



Grau en Enginyeria de Vehicles Aeroespacials

Title:

***Optimization study of dynamic vibration
absorbers parameters and distribution for its
application on railway tunnels for the reduction
of railway-induced vibration***

Document content: REPORT

Delivery date: 12/06/2015

Author: Víctor Cubría Radío

Director: Robert Arcos Villamarín

Abstract

This study develops an algorithm in order to obtain the optimal distribution of Dynamic Vibration Absorbers (DVAs) in underground railway infrastructures for the reduction of railway-induced vibrations. It has been developed and formulated the railway-induced vibration problem to obtain the vibrational response of the system. With this theoretical background a genetic algorithm (GA) has been implemented and refined; with the aim of saving time some processes have been applied to the GA, parallel processing, as well as vectorization of the code are the most successful methods. Furthermore, to release computing charge of the GA, a hybrid algorithm has been proposed, combining the GA with a pattern search algorithm to improve the accuracy of the variables of type double. Finally, to ensure that the solutions obtained correspond to global minimums, it is proposed a testing model using different software, OPTI Toolbox Studio to check the solutions.

Contents

LIST OF FIGURES	IV
LIST OF TABLES	V
1. AIM	1
2. SCOPE	2
3. REQUIREMENTS	4
4. JUSTIFICATION	5
5. STATE OF THE ART	6
5.1 Railway induced vibration problem	8
5.2 Optimization solving problem	10
6. SOFTWARE	11
7. THEORETICAL BACKGROUND	13
7.1 Mitigation of Train-induce Ground Vibration	13
7.2 Methodology	14
7.2.1 Tunnel-soil interface displacement calculation	14
7.2.2 Equivalent forces calculation	15
7.2.3 Far-field displacement calculation	15
7.2.4 Coupled soil-tunnel-DVA system	15
8. MIXED INTEGER OPTIMIZATION	22
8.1 Genetic algorithm	24
8.1.1 Overview of genetic algorithm	24
8.1.2 Fitness function	28
8.1.2.1 Step 1: Generate DVASET	29

LIST OF FIGURES

8.1.2.2	Step 2: Call H_UnitaryExteranlForce_E1.m	30
8.1.2.3	Step 3: ISO – 2631-2 Application	32
8.1.3	Configuration of the GA	33
8.1.4	Performance improving methods	36
8.1.4.1	Vectorization of the code	36
8.1.4.2	Parallel processing	37
8.1.4.3	Study of parallel pool configuration	38
8.1.4.4	Refinement of the GA	42
8.1.5	Improving accuracy using pattern search	45
8.1.6	Global minimum checking method	46
9.	RESULTS AND ANALYSIS	49
10.	ENVIRONMENTAL AND SAFETY IMPLICATIONS	56
10.1	Acoustic reduction	56
10.2	Building damage reduction	57
10.3	Energy cost reduction	59
11.	FUTURE LINES OF INVESTIGATION	60
12.	TIMELINE	61
13.	BUDGET	64
14.	ECONOMIC VIABILITY	65
15.	CONCLUSIONS	67
16.	BIBLIOGRAPHY	69

List of Figures

5.1	Schematic overview of implications of noise and vibration emitted from railway, image from [1].	6
5.2	The three railway options in which vibration and noise are generated. Image from [2].	7
5.3	Detailing of the DTRD, image from [3].	8
5.4	Detailing of the DVA model fixed on a single cross sectioned track. Mesh top view and mesh bottom view (using Ansys 11). Image from [4].	9
7.1	Two views of tunnel in full-space with one distribution of DVAs, isometric view (a) and front view (b).	16
7.2	Dynamic vibration absorber behaviour. Extracted from [5].	16
7.3	Relation between external forces and virtual forces (a) and (b) shows that when force F rotates, the displacements will also rotate in the same direction. Extracted from [5].	19
8.1	Schematic diagram of how the main algorithm works for one generation running in parallel three fitness function.	23
8.2	Figure 8.2: Schematic diagram which illustrates the three types of children. Extracted from [6].	27
8.3	Example of the output of H_UnitaryExternalForce_E1.m.	31
8.4	Schematic view of how parallel pool works. It is only possible to have one parallel pool accessible at time from a MATLAB client session. Image from [7].	39
9.1	Graph of the square of the acceleration against frequency, for three cases: without DVAs, for one DVA and for ten.	50
10.1	Transient vibration guide values for damage, extracted BS 7385: Part 2 [8].	58
12.1	Gantt chart of the future study. It has been considered that it starts on September 18 th and finishes on January 1 st .	63
14.1	Position of each distribution of DVAs. xNode and yNode correspond to the coordinates of the Evaluators (see annex B) .	65

List of Tables

8.1	Performance comparison classical algorithms and genetic algorithms.	25
8.2	Example of possible parameters used to generate DVAs.	29
8.3	Formulas for generating the transfer function with the corresponding values of f_1 , f_2 and f_3 .	32
8.4	Inputs of the GA.	34
8.5	Configuration options of the GA.	35
8.6	Comparative of the initial version of the code and the vectorized form with an extract of the code in which the normative is applied.	36
8.7	Computing time obtained for each parallel pool configuration and iterations.	41
8.8	Best time values, mean time values and average values. In green the two best options.	42
8.9	GA parameters for the optimization of the case of two DVAs.	43
8.10	Example of the strategy followed by the algorithm for the case of two DVAs.	44
8.11	Solutions of the solver NOMAD for the optimization for one DVA.	46
9.1	Solutions from 1 to 6 DVAs.	52
9.2	Solutions from 7 to 9 DVAs.	53
9.3	Solutions for 10 DVAs.	54
10.1	Examples of further solutions being implemented in various countries of the EU [9].	57
14.1	Total cost of the materials and components of each distribution of DVAs.	66

1. Aim

The aim of this study is to develop an algorithm able to obtain the optimal distribution of Dynamic Vibration Absorbers (DVAs) in underground railway infrastructures for reduction of railway-induced vibrations.

2. Scope

This study develops an algorithm with the objective to obtain an optimal distribution of DVAs, for a specific case study, in underground railway infrastructures for reduction of railway-induced vibrations. Then the scope of the project is limited to the application of these devices, DVAs, on the tunnel's walls.

According to these limitations, several important points or tasks have been fixed, which have been developed along the study, to finally achieve the objectives. In this way, the study has been structured in three primary parts. In the first one, theoretical concepts have been compiled and later developed to compose the theoretical background over the final algorithm will be developed:

- Learn about the basic concepts of the railway-induced vibration problem.
- Study of how to couple DVAs to tunnel walls and the assembly process of the final situation with the DVAs incorporated to the system, so finally evaluate the vibrational response.
- Learn about metaheuristic models for optimization problems, focusing in understanding of genetic algorithms. As a part of the genetic algorithm review stage, also learn to use Optimization Toolbox included in MATLAB's software.
- Research for alternatives processing methods, to obtain a more accurate result using a hybrid algorithm or just finding the global minimum value with other software as OPTI Toolbox.

Once established and clearly defined the theoretical background of the study, it is possible to proceed to the practical development. In this second part, the entire processes and investigation methodologies for the obtaining of optimum DVAs position:

- Develop an algorithm for obtaining the vibration levels in the building due to the train passage. The functions which develop this part have been provided by the LEAM (the research center in which this study has been developed).

- Develop a genetic algorithm able to find the optimal distribution and parameters of the DVA in terms of the reduction of vibration.
- Analyse and develop, if it is necessary, a hybrid algorithm integration or combination and also check results with an alternative one as OPTI Toolbox.
- Test the code for each specific case. As a first approach it will be developed the optimization study for each possible number of DVAs installed.

Finally a third section, which includes testing of the final code as well as a complementary part, evaluating the cost associated to the implementation of the algorithm. Furthermore following this evaluation it will be presented an economic viability study, including the budget and also a planning and scheduling for future lines of investigation.

3. Requirements

The list of requirements that this study has to meet in order to be considered successful is the following:

- A.** To be able to develop theoretical concepts, applied to the railway-induced vibration problem and the obtaining of the vibration levels in the building, to the algorithm.
- B.** To develop an algorithm able to obtain the vibration levels in the building due to the train passage and assembly it, efficiently, in the genetic algorithm.
- C.** To apply successfully the theoretical background related to the genetic algorithm into the development of the code and also working with the Global Optimization Toolbox of MATLAB for the implementation of the genetic algorithms.
- D.** To adapt the algorithm for a global optimization case study and to be able to achieve a solution for this optimization problem.
- E.** To develop an optimization algorithm computationally cheap enough to obtain a good solution of the case study.

4. Justification

Nowadays subway and metro are commonly used as a means of transport by most people. While train is used for mid-long journeys, subway circulates covering short distances. It is in this second part where it is easier to observe the problem which this study tries to solve, or at least reduce it as much as possible. This problem is about perturbations, by way of vibrations to which near buildings to the focus of emission are subjected. These vibrations are a direct consequence of the train passage. This fact generates a state of discomfort in the affected neighbourhood, for example a recent case happened in Valorio (Zamora) where the passage of the train produces noise and vibrations affecting 200 people [10].

Reducing vibration is a topic that has been treated deeply along the expansion and using of railway. Normally these problems are associated to the train or to the track and involve big changes in the structure. So trying to avoid this situation is how DVAs implementation takes advantage. This type of devices is very useful in complex systems as railway tunnels. The justification consists in how these DVAs are able to improve the dynamic behaviour without making any relevant modifications. Furthermore acoustic reduction will be achieved as a consequence of vibration absorbing.

On the other hand, there is another need, an algorithm able to find the optimal distribution and parameters of the DVA in terms of the reduction of vibration. But it is not only about generating an algorithm there is an economic need, the algorithm has to be computationally cheap enough to obtain a good solution of the case study. In this way the justification of why using genetic algorithm is presented. Genetic algorithm (GA) works as a metaheuristic method solving optimization problems. GA offers the possibility to be refined and easily to save time, and of course money.

At the end, the justification of this study is the need of providing an alternative solution (regarding to the conventional methods that will be clearly explained in next chapter), of railway-induced vibrational response in railway tunnels; and as fundamental aspect, assemble this response to a genetic algorithm computationally cheap enough to allow the economic viability for its implementation.

5. State of the art

Vibration reducing and control is one the topics, associated with the railway system, in which research groups and private corporate are usually focused on. There are other means of transport like boats and planes, in which a noise and vibration engineer can work, but railways are unique in the challenges they bring to professionals involved in the fields of noise and vibration. There are three differentiated aspects, studies normally focus on:

- **Environmental** consequences.
- **Safety** and **comfort** of the passenger and employee.
- Design of the **infrastructure**.

These three aspects affect directly to the customer (see clearly in figure 5.1), so exist legislative and commercial pressures demanding on a modern railway. Methods for dealing with these demands have changed along years of research work.

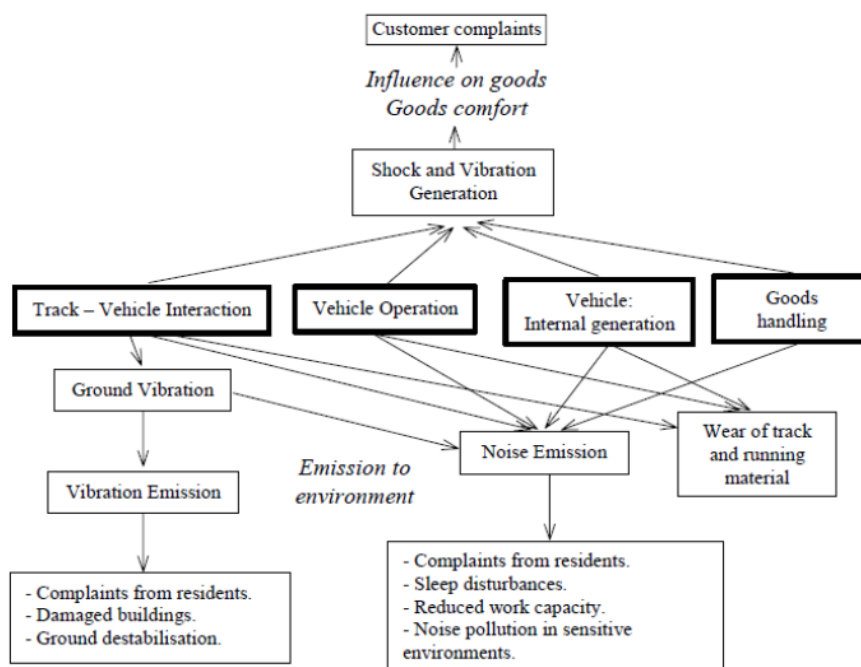


Figure 5.1: Schematic overview of implications of noise and vibration emitted from railway, image from [1].

Approaching to vibration and noise problem, both elements are analysed as a **wave phenomenon**. Following there is a definition of each term:

- **Noise**: is defined as sound waves propagating through the **air**.
- **Vibration**: is defined as waves propagating through the **ground**.

Both elements appear as a result of oscillations of wheels and rails due vehicles rolling on tracks, i.e. dynamic forces arising due to wheel-rail interference roughness. Working at high frequencies, the excitation energy expands through the air as sound waves, noise, whereas lower frequency waves transmit from the rail to the lower parts of the track structure, through the ground and to the objects in the ground. Typically frequency range in which vibrations and structure-born noise occur is **0-100 Hz** and noise 30-2000 Hz [2].

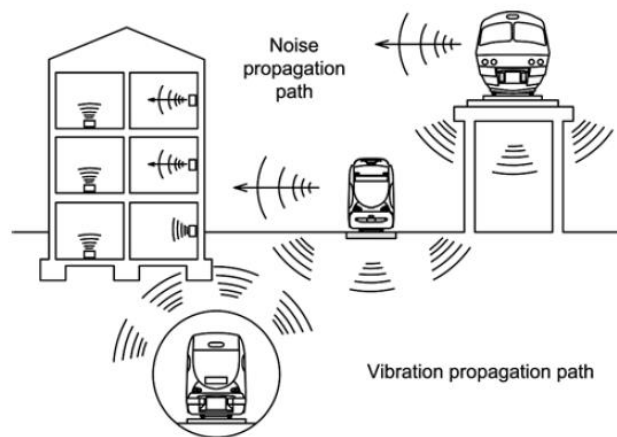


Figure 5.2: The three railway options in which vibration and noise are generated. Image from [2].

As it is possible to see in figure 5.2, there are three main railway variations which generate vibration and noise. This study is focused on the railway tunnel application. Inside the building it is possible to appreciate some devices, called **evaluators**. These devices are useful in order to obtain vibration and noise level that buildings receive. Furthermore, a lot of review treating mitigation measures has been done along last years, there are two interesting ones: constructing barriers and placing track in tunnels, allows changing the propagation path, and the installation of elastomeric pads under the track construction, following this idea, the most effective mitigation measure in this aspect would be elastically embedding the rails. The main problem of the first option is the economical aspect, high construction costs and also the lack of space for their positioning in densely built urban areas. In the opposite side, the second measure can easily be implemented during regular maintenance or reconstruction of rail tracks.

5.1 Railway induced vibration problem

Through the years railway induced vibration has been growing as a matter of environmental concern, in parallel means of transport have evolved since the first days, increasing of vehicles speeds and weight have as consequence higher vibration level. So there is a need to lead with structural damages to nearby buildings and the noise generated. To solve this problem normally there exist conventional procedures which require infrastructure modifications, changing the construction lines and placing beds under the ballasts and the majority only act on a specific part of the dynamic behaviour of the track. To avoid this situation, a new method using **dynamic vibration absorbers** (DVAs), has been introduced.

In this line of investigation, some interesting publications have been published about DVAs applied to solve the railway-induced vibration. Then, most relevant are going to be resumed:

- *A double tuned rail damper-increased damping at the two first pinned-pinned frequencies*: in this paper J. Maes *et al.* present a possible solution to reduce the noise generated by the pinned-pinned frequencies by providing a DVA. Pinned-pinned frequencies are referred to standing waves whose nodes are positioned exactly at the sleeper supports. To attenuate the vibrations, a DVA has been developed, called **Double Tuned Rail Damper** (DTRD) [3].

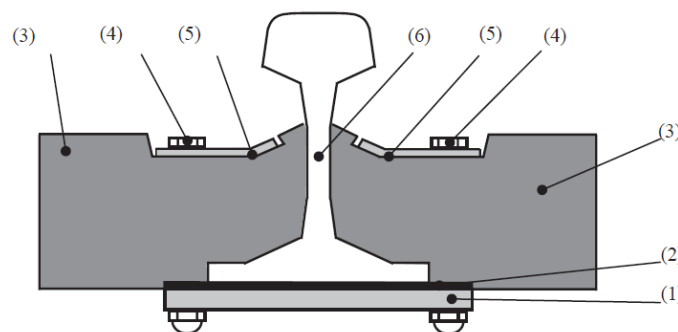


Figure 5.3: Detailing of the DTRD, image from [3].

The DTRD consists of two major parts: a **steel plate** (1), which is connected to the rail with an interface of an **elastic layer** (2), and two **rubber blocks** (3). Two **bolts** (4) and **metal plates** (5), actuate pushing the rubber blocks against the **rail** (6).

As a result, the model showed considerable attenuation envisaged pinned-pinned frequencies. But there were some problems of the DVA's behaviour, as solution, they decided to measure the exact pinned-pinned frequencies of the track in which the damper will be applied.

- *Design and analysis of dynamic vibration absorber (DVA) to reduce vibration in rails:* S. Naresh kumar *et al.* present a new DVA system design, including simple clamping technique with the rail along the line. The advantage of this study compared with others is that the replacement of the device can be done easily in case of failure [4].

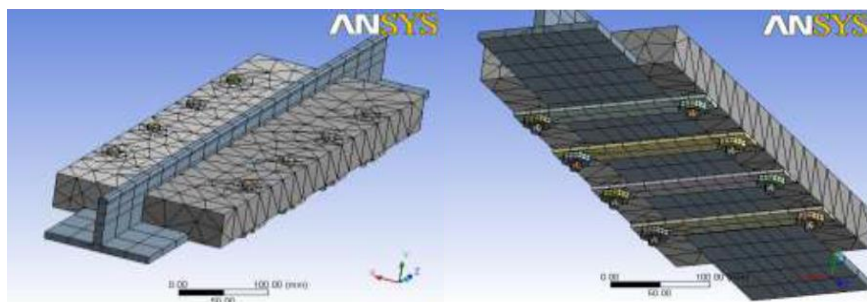


Figure 5.4: Detailing of the DVA model fixed on a single cross sectioned track. Mesh top view and mesh bottom view (using Ansys 11). Image from [4].

The final conclusions are that the DVA proposed reduces the vibration comparatively high rate than others by the collected details. These details consist of specific properties of the parameters of the materials used in it. At the end, the high frequency vibrations are reduced to their maximum extent that they occur below the noise level.

About DVAs application and improvement there are other interesting contributions: some studies have focused on improving DVA's architecture, in order to avoid internal resonance [11]. Others proposed a complete redesign of DVA's architecture [12] and there are also papers which study vibrational effects and variations as a consequence of the different characteristics parameters of the DVA [13].

The idea is that all previous studies have oriented their solutions applying devices or solutions over the rail or the track, but in this project DVAs are applied on the **railway tunnels**, and to this day there is no literature facing this alternative solution.

5.2 Optimization solving problem

As a final objective, the study proceeds to develop an optimization algorithm able to find the optimum positions of DVAs which minimize the vibration levels received by the evaluators.

In order to solve the optimization problem a genetic algorithm has been implemented. A genetic algorithm, in computer science, is referred as a **metaheuristic model**: a high-level procedure designed to find, generate, or select a lower-level procedure of heuristic, partial search algorithm that may provide a sufficiently good solution to an optimization problem. Genetic algorithms use a type of search strategy which has a learning component to the search [14].

There are some works following this line. One interesting approach consists on the development of optimization with genetic algorithms focusing on parameters of active vibration control [15]. An alternative one has used Non Nominated Genetic Algorithms to optimize the characteristics of the DVAs themselves [16]. There are also different interesting approaches to the optimization problem, for example A. Kaveh and S. Malakouti Rad [17] propose using a hybrid genetic algorithm and particle swarm optimization for the force method-based simultaneous analysis.

For the algorithm developed in this study, it has been useful to learn from the work developed by E. Matas [18] and D. Sellés [19] in their respective projects. In those two works, are presented an study developing a methodology allowing to linearly couple the dynamic response of a three dimensional structure with vibration absorbing elements and to adapt it to an optimization cycle, in the other is developed a process which allows to optimize the number, position or dynamic characteristics of vibration absorbing devices that will be placed in airplane structural elements in order to achieve a vibration reduction. They developed a general optimization algorithm with their own code which has been really useful for this study.

6. Software

Along the study there are two differentiated parts where the computation task has been developed:

- **Frequency response model:** obtaining the **vibration levels** in the building due to the train passage; to run the optimization algorithm is required a function which computes a matrix (H_{DVA}), and includes the needed values and parameters of the DVAs and the rest of the system.
- **Genetic algorithm implementation:** a critical point of the algorithm is the design and programming of the fitness function. The fitness function corresponds to the stage of the algorithm which relates the inputs (H_{DVA} , DVAs' values and parameters, etc.) with the vibration levels. So most of the efforts dedicated to the genetic algorithm are focused on the fitness function.

These two stages are main computing parts of the study and have been programmed using **MATLAB** (R2014a):

- **MATLAB** is a high-level language and interactive environment used by millions of engineers and scientists worldwide; it allows exploring and visualizing ideas and collaborating across disciplines including signal and image processing, communications, control systems, and computational finance. Developed by *MathWorks*, American company, leading developer of mathematical computing software for engineers and scientists [20].

The choice of MATLAB as the platform responds to several reasons, it is a relatively simple software, there is a lot of background with guides to learning using it and offers a programming language efficient and adapted to engineering tasks. As a good point there is also the fact that allows readily to process, export and import text files, very useful in terms to work with complex algorithms with many inputs and variables to manage. Finally, because MATLAB is one of the software used by the department where this study has been developed, and because its accessibility through university licenses.

Specially, two modules of MATLAB have been used:

- **Global Optimization Toolbox:** provides methods that search for global solutions to problems that contain multiple maxima or minima. It includes global search, multistart, pattern search, **genetic algorithm**, and simulated annealing solvers [6].
- **Parallel Computing Toolbox:** allows to solve computationally and data-intensive problems using multicore processors, GPUs, and computer clusters. High-level constructs—parallel for-loops, special array types, and parallelized numerical algorithms—allow parallelizing MATLAB applications without Compute Unified Device Architecture (CUDA) or Message Passing Interface (MPI) programming [21].

Finally, it has been proposed a model to check the results and ensure that solution is global minimum. With this purpose it has been used **OPTI Toolbox**:

- **OPTI Toolbox:** is a free MATLAB toolbox for constructing and solving linear, nonlinear, continuous and discrete optimization problems [22].

7. Theoretical background

As it has been explained before, the aim of this study is to develop an algorithm able to obtain the optimal distribution of Dynamic Vibration Absorbers (DVAs) in underground railway infrastructures for reduction of railway-induced vibrations. To achieve this goal it is needed to develop a thorough theoretical study of the vibrational response of the system to be able to generate, in second term, the code.

In this chapter it will be shown all the theoretical background needed to cover the evaluation of the vibration along the system. This part, in the code, will appear in each fitness function (to solve the optimization problem for each number of DVAs) and in the function `H_UnitaryExternalForce_E1.m`, all this stuff will be explained in section 8.

7.1 Mitigation of Train-induce Ground Vibration

As the vibrations that are induced by underground railways propagates through the soil and can be sensed directly as vibration or indirectly as re-radiated noise that causes serious annoyance for occupants, modelling the vibration of underground railways is gaining more interests in order to develop countermeasure to mitigate the vibration. Therefore, in this study, it is intended to attenuate the ground vibration from underground railways by the use of **Dynamic Vibration Absorbers** (DVA) as innovative countermeasures.

In order to reach the aim, first of all, the surface response due to harmonic point load that is applied in the tunnel surface has to be defined. There are two popular methods to model the train-induce ground vibration, **Pipe-in-Pipe** (PiP) and **finite element-boundary element** (FE-BE) model. The former is based in using two concentric pipes, where inner pipe and outer pipe, with infinite outer radius, represent tunnel and soil, respectively [23], [24]. The latter, employs finite element to model the tunnel wall and boundary element to model surrounding soil and by using the coupled FE-BE, the surface response can be found [25].

In the case of a tunnel embedded in a full-space, the PiP model has been validated against the coupled FE-BE model [5]. Not only a good agreement but also more computational efficiency from the PiP model is achieved. While the surface response is

specified, the second part of the study will be concerned with the application of DVAs to mitigate the vibration generated by underground railways. In this part, the DVAs are placed on the tunnel surface and by enforcing continuity of deflection in the position of DVAs; the receptance of the system due to DVAs can be achieved. Therefore, by considering the receptance of the system in the absence of DVAs, which is found in this first part, their efficiency in vibration mitigation can be evaluated.

7.2 Methodology

The model that will be presented in the following is based on the assumption that near field displacement of the tunnel is not influenced by the existence of free surface. The methodology which is employed to evaluate the efficiency of DVA consists of four steps that are explained in following sections:

1. Tunnel-soil interface displacement calculation.
2. Equivalent forces calculation.
3. Far-field displacement calculation.
4. Coupled soil-tunnel-DVA system.

By following each one of these steps, it will be possible to obtain the response of the system. This response will be represented as a variable which will be taken as the parameter to minimize in the algorithm.

7.2.1 Tunnel-soil interface displacement calculation

In order to compute the response of the tunnel embedded in a full-space due to a vertical point load the dynamic interaction between the tunnel wall and a full-space with a cylindrical cavity (PiP model) is employed where it is assumed that near-field displacement of the tunnel is not influenced by the existence of free surface. This model is completely explained in the work of Forrest [26] which is also employed in this investigation. Briefly, it is composed of three stages, the **displacement and stress solutions** while the tunnel is modelled as a cylindrical shell of infinite length, determining the **wave propagation** in a full-space with a cylindrical cavity and solution of the **coupled tunnel-soil interaction** problem. It should be noted that the model is not variant in one direction; therefore, all the calculations are done in frequency-wavenumber domain. In other words, all the displacement will be achieved as $\hat{\tilde{u}}_i(k_x, y, z, w)$, where *tilde* and *hat* represents wavenumber and frequency domain, respectively and $i = r, \theta, x$.

7.2.2 Equivalent forces calculation

In this step, a model of a full-space without the tunnel is considered and a number of loads, that can produce the same displacements as ones which are calculated in previous stage, are determined. This problem can be solved by use of fundamental solution that relates the displacement and stress at some location in an elastic body caused by dynamic sources placed elsewhere in the medium or **2.5D green function** for elastodynamic problem for full-space which is presented by Tadeu *et al.* [27].

7.2.3 Far-field displacement calculation

The surface response due to the unitary harmonic force at tunnel surface can be determined by applying the set of equivalent forces which are calculated in subsection 7.2.2. As the relation between the forces and displacements in the elastodynamic media is needed, by employing the green function for 2.5D half-space presented by Tadeu *et al.* [27], the surface displacements resulting from equivalent forces can be found.

By following these three steps the receptance of the tunnel embedded in the half-space due to unitary harmonic force can be found. Later on, in the fourth step, the receptance of the system due to unitary harmonic external force is defined and by forcing the system to have the same displacement in the DVA position when external force and DVA forces are applied a system of equation is achieved where the DVA forces are unknowns. By solving the systems of equations, the receptance of the system in the presence of DVAs can be derived.

7.2.4 Coupled soil-tunnel-DVA system

In this section the methodology that is used to couple DVAs to the tunnel surface in order to find DVAs forces will be explained. As it can be seen in figure 7.1, in the first step to achieve better understanding of the problem, just one periodic distribution of DVAs is considered and it is assumed that all the DVAs in the distribution have the same mass, stiffness and damping.

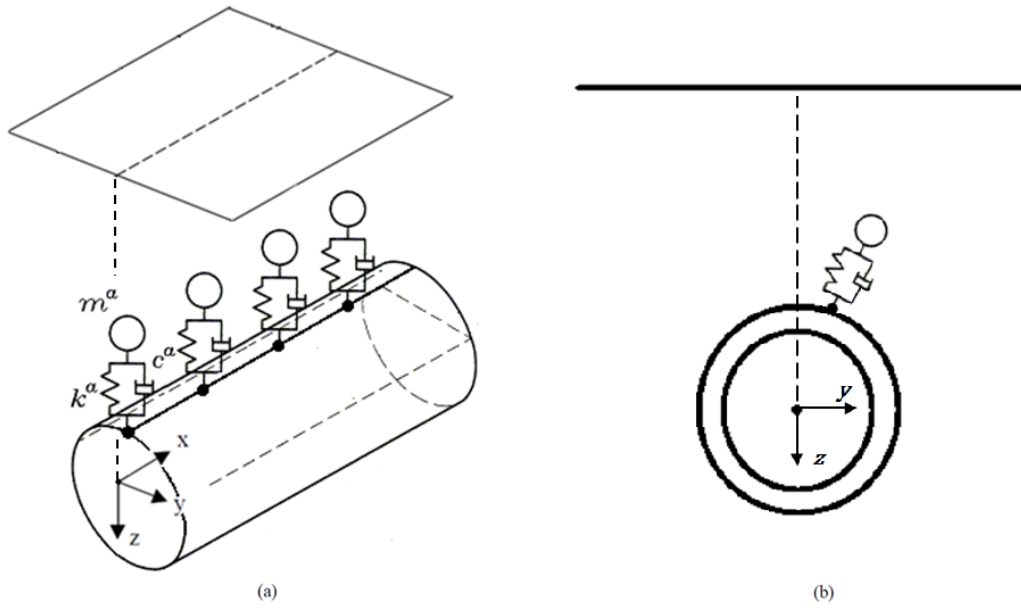


Figure 7.1: Two views of tunnel in full-space with one distribution of DVAs, isometric view (a) and front view (b).

By assuming that the forces are applied in the same direction, the applied forces to the system are:

$$f(x, t) = P(t)\delta(x) + \sum_{n=-\infty}^{+\infty} \left(k + c \frac{\partial}{\partial t} \right) (u_n^{\text{DVA}} - u_n) \delta(x - x_n) \quad (7.1)$$

where k and c are **stiffness** and **viscous damping** of DVA, respectively, $x_n = nL$ where L is the distance between each DVA, u_n is the displacement of the system in the position and direction of the n^{th} DVA, u_n^{DVA} is the displacement of the center of the mass of the n^{th} DVA. Figure 7.2 shows the behaviour of n^{th} DVA, where

$$-\left(k + c \frac{\partial}{\partial t} \right) (u_n^{\text{DVA}} - u_n) = m \ddot{u}_n^{\text{DVA}} \quad (7.2)$$

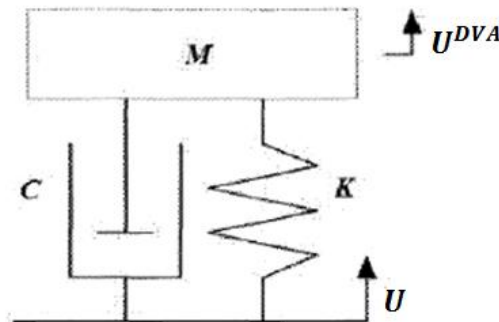


Figure 7.2: Dynamic vibration absorber behaviour. Extracted from [5].

By taking the advantages of **invariant coupled soil-tunnel system** along the direction of the rails and working in frequency-wavenumber domain, the computational costs can be decreased greatly. As a result, by considering the definition of Fourier Transform as:

$$\hat{f}(k_x, \omega) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, t) e^{-i(\omega t - k_x x)} dx dt \quad (7.3)$$

and applying on equations 7.1 and 7.2, those equations can be found in frequency-wavenumber domain:

$$\hat{f}(k_x, \omega) = P + \sum_{n=-\infty}^{+\infty} (k + i\omega c)(u_n^{\text{DVA}} - u_n) e^{i(k_x x_n)} \quad (7.4)$$

where tilde and hat represent wavenumber and frequency domain respectively, and

$$-(k + i\omega c)(u_n^{\text{DVA}} - u_n) = -\omega^2 m u_n^{\text{DVA}} \quad (7.5)$$

The relation between displacements of mass center of DVA, u_n^{DVA} , and the displacement of attaching position to the tunnel, u_n , can be found by rewriting the latter as

$$u_n^{\text{DVA}} = \frac{k + i\omega c}{k + i\omega c - \omega^2 m} u_n \quad (7.6)$$

Replacing eq. 7.6 in Eq. 7.4:

$$\hat{f}(k_x, \omega) = P + \sum_{n=-\infty}^{+\infty} K(u_n) e^{i(k_x x_n)} \quad (7.7)$$

where the first and second terms in right and are external force and DVAs force, respectively and

$$K = (k + i\omega c) \left(\frac{k + i\omega c}{k + i\omega c - \omega^2 m} - 1 \right) \quad (7.8)$$

As the forces are defined in frequency-wavenumber domain, by multiplying them to their related receptances (which will be explained later and can be derived by following steps 7.1.2.1, 7.1.2.2 and 7.1.2.3) and applying the Inverse Fourier Transform that is defined as

$$\hat{u}(x, \omega) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{u}(k_x, \omega) e^{-i(k_x x)} dk_x \quad (7.9)$$

The response of the system can be written as

$$u = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \widehat{H}(k_x, \omega) P e^{-i(k_x x_n)} dk_x + \frac{1}{2\pi} \int_{-\infty}^{+\infty} \widehat{H}_1(k_x, \omega) \left[\sum_{n=-\infty}^{+\infty} K u_n e^{i(k_x x_n)} \right] e^{-i(k_x x_n)} dk_x \quad (7.10)$$

where the \widehat{H} is the receptance of the system due to unitary external force and \widehat{H}_1 is the receptance of the system due to unitary DVA forces. The equation 7.10 can be rewritten as

$$u \cdot n^{\text{DVA}} = P \widehat{H} \cdot n^{\text{DVA}} + \sum_{n=-\infty}^{+\infty} [\widehat{H}_1(x - x_n, \omega) \cdot n^{\text{DVA}}] K u_n \quad (7.11)$$

or also by

$$u_t = \widehat{H}(x_t, \omega) + \sum_{n=-\infty}^{+\infty} [\widehat{H}_1(x_t - x_n, \omega)] K u_n \quad \text{and } t = 1, 2, \dots, N \quad (7.12)$$

where N is the number of DVAs along the tunnel direction, which will be found in the optimization process. By approximating the summa with the finite number of equation and as the response of the system owing to unitary external force is being sought, a set of system of equations can be made as

$$\begin{cases} u_1 = \widehat{H}(x_1, \omega) + \sum_{n=1}^N [\widehat{H}_1(x_1 - x_n, \omega)] K u_n \\ \vdots \\ u_N = \widehat{H}(x_N, \omega) + \sum_{n=1}^N [\widehat{H}_1(x_N - x_n, \omega)] K u_n \end{cases} \quad (7.13)$$

Where the displacements of the system in the position of DVAs, \widehat{H} and \widehat{H}_1 are unknowns. \widehat{H} is the tunnel-soil interface displacement due to unitary harmonic force at the tunnel surface that is studied in the section 7.1.2.1. As in the section 7.1.2.1, it is assumed that the near-field displacement of the tunnel is not influenced by the existence of free surface, in other words, the soil-tunnel interface displacement calculation is in the full-space which makes the problem symmetric, therefore, as it can be seen in figure 7.3 by rotating the position of the force, the resulting displacement will rotate. As a result, the \widehat{H}_1 can be found by reshaping the \widehat{H} according to the position of DVA. Consequently, we have N equation with N unknowns. By solving this set of

7. Theoretical Background

equations of motion, the displacements of the system at DVA positions can be found that simply result in DVAs forces by knowing the stiffness of absorbers.

In the last part, finding the response of the system due to unitary harmonic force at surface in the presence of DVAs is considered as the final intention which is the summation of the response of the system due to external force and DVAs forces. The former is found in the section 7.1.2.3; the latter also can be calculated by following three steps in subsections 7.1.2.1, 7.1.2.2 and 7.1.2.3. As it was mentioned previously, the soil-tunnel interface displacement calculation (section 7.1.2.1) is done in the full-space which makes the problem symmetric; therefore, by rotating the position of the external force, the resulting virtual forces will rotate. In other words, according to the position of unitary force at tunnel surface, just the position of virtual forces will be changed. Consequently, the results of two first subsections 7.1.2.1 and 7.1.2.2 can be used to find the equivalent virtual forces resulting from DVAs unitary forces.

Afterwards, by employing 2.5D green function half-space presented by Tadeu *et al.* [27], the surface response due to unitary DVAs forces can be evaluated. The final response of the system is the summation of response of the system due to unitary external force and DVA forces.

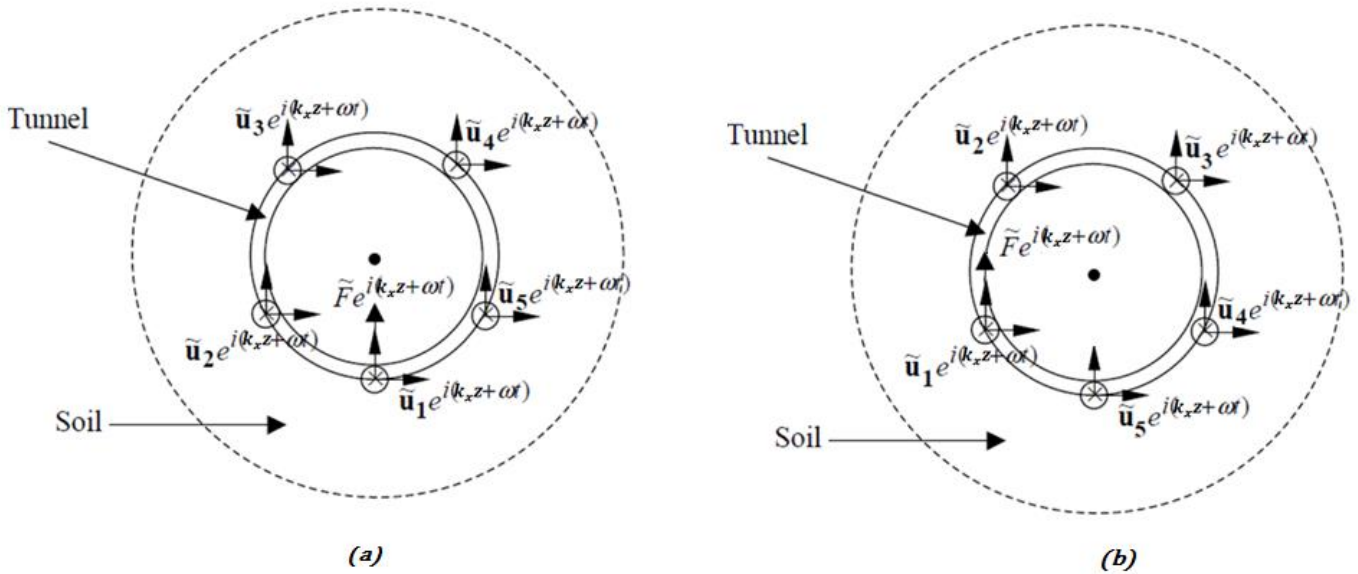


Figure 7.3: Relation between external forces and virtual forces (a) and (b) shows that when force F rotates, the displacements will also rotate in the same direction. Extracted from [5].

In the case of more than one distribution of DVAs, the same steps should be implemented. The summation in equation 7.12 shows the effect of all N absorber on the c^{th} DVA, by increasing the number of distributions, the effect of other distribution

on c^{th} DVA also needs to be observed. Therefore, by considering M distribution, the Eq. 12 is changed to

$$u_{s_l} = \hat{H}(x_{s_l}, \omega) + \sum_{m=1}^M \sum_{n=1}^N [\hat{H}_m(x_{s_l} - x_n, \omega)] K u_{m_n} \quad (7.14)$$

where

$$s = 1, 2, \dots, M \text{ and } l = 1, 2, \dots, N \quad (7.15)$$

In other words, s represents the number of distribution in tunnel cross section and l represents the number of DVAs in each distribution. By expanding equation 7.14, the following set of equations can be found:

$$\left\{ \begin{array}{l} u_{1_1} = \hat{H}(x_{1_1}, \omega) + \sum_{m=1}^M \sum_{n=1}^N [\hat{H}_m(x_{1_1} - x_n, \omega)] K u_{m_n} \\ \vdots \\ u_{1_N} = \hat{H}(x_{1_N}, \omega) + \sum_{m=1}^M \sum_{n=1}^N [\hat{H}_m(x_{1_N} - x_n, \omega)] K u_{m_n} \\ u_{2_1} = \hat{H}(x_{2_1}, \omega) + \sum_{m=1}^M \sum_{n=1}^N [\hat{H}_m(x_{2_1} - x_n, \omega)] K u_{m_n} \\ \vdots \\ u_{2_N} = \hat{H}(x_{2_N}, \omega) + \sum_{m=1}^M \sum_{n=1}^N [\hat{H}_m(x_{2_N} - x_n, \omega)] K u_{m_n} \\ \vdots \\ u_{M_1} = \hat{H}(x_{M_1}, \omega) + \sum_{m=1}^M \sum_{n=1}^N [\hat{H}_m(x_{M_1} - x_n, \omega)] K u_{m_n} \\ \vdots \\ u_{M_N} = \hat{H}(x_{M_N}, \omega) + \sum_{m=1}^M \sum_{n=1}^N [\hat{H}_m(x_{M_N} - x_n, \omega)] K u_{m_n} \end{array} \right. \quad (7.16)$$

As it can be seen, there are $M \times N$ unknowns and $M \times N$ equations, which finally result in displacements at DVAs positions, by knowing the stiffness of DVAs, their forces can be found easily. Finally, the surface response of the system can be found in frequency-

7. Theoretical Background

spatial domain which can be transferred to time-spatial domain by applying another Inverse Fourier Transform, like

$$H(x, t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{H}(x, \omega) e^{i(\omega t)} d\omega \quad (7.17)$$

8. Mixed integer optimization

After developing all the theoretical concepts, it is time to evaluate which type of optimization problem has to be solved. To understand the type of problem, it is required an overview of how the general algorithm works in terms of inputs and outputs; inputs are introduced in the algorithm using the DVAParameters.m function while output comes from H_UnitaryExternalForce_E1.m. So the main parts or components of the algorithm are:

- **Genetic algorithm.**
 - **Fitness function.**
 - DVAParameters.m.
 - H_UnitaryExternalForce_E1.m.
- **— Pattern search —** this part has been included as an optional part to be implemented by the user if it is needed. It is another solver which runs after the GA to improve the accuracy of the variables of type double obtained from the GA, see section

DVAParameters.m and H_UnitaryExternalForce_E1.m are included inside the fitness function, but they have been remarked as main components because they play an important part of the work. The first function gives all the parameters which characterized the DVAs and this data is managed as an input for the second function which as a result gives a 3D matrix used to obtain the vibrational response of the system. Those functions have been supplied by the LEAM (the research center in which this study has been developed).

In figure 8.1 there is a schematic diagram of how the GA works. The idea is that after calling the GA, immediately the software starts to generate the first generation. To speed up this process, the individual is computed in parallel (see subsection 8.1.4.2), that means that in parallel, first DVAParameters.m and then H_UnitaryExternalForce_E1.m are run to obtain the fitness value. From all the fitness values are selected the best ones (**elite individuals**). The schema represents a parallel processing configuration of three workers, but as it will be explained later on, it is possible to increase the number of worker to improve the performance of the GA. This process is run iteratively by using some evolutionary strategies to obtain the best fitness value.

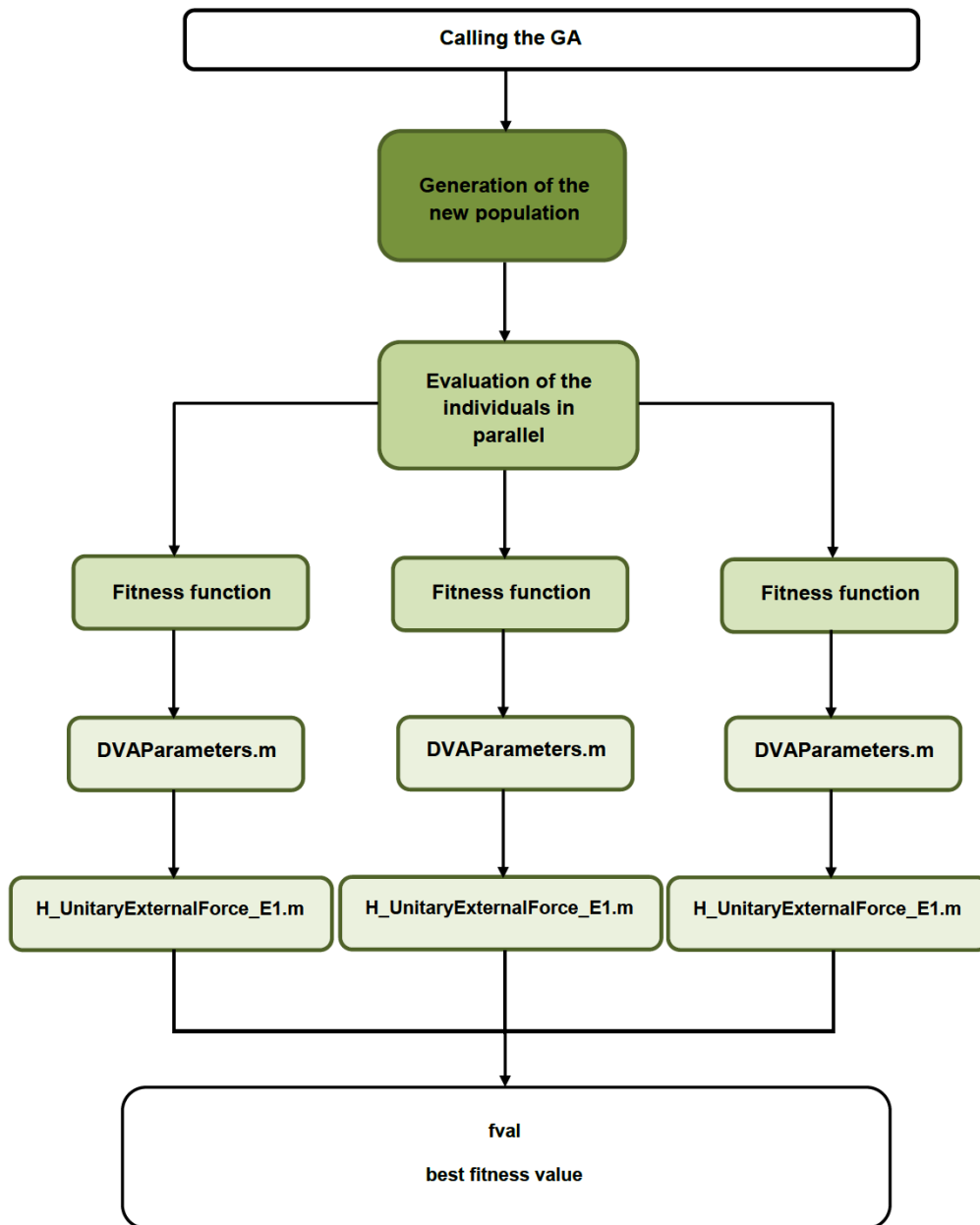


Figure 8.1: Schematic diagram of how the main algorithm works for one generation running in parallel three fitness function.

An important topic is to evaluate the inputs of the algorithm because these values define which type of problem is and in this way which solver has to be implemented. In next chapter inputs of the GA will be explained deeper by analysing DVAParameters.m function. In any case, for the moment, what has to be clear is that some of the inputs are of the type **integer** and others of type **double**. This involves that the problem to solve is a **mixed integer optimization** problem. That is the reason by using GA as solver, because GA can solve problems when certain variables are integer-valued.

8. Mixed integer optimization

Anyway restrictions exist on the types of problems that GA can solve with integer variables:

- No linear equality constraints.
- No nonlinear equality constraints.
- Only `doubleVector` population type.
- No custom creation function (`CreationFcn` option), crossover function (`CrossoverFcn` option), mutation function (`MutationFcn` option), or initial scores (`InitialScores` option).
- GA uses only the binary tournament selection function (`SelectionFcn` option), and overrides any other setting.
- No hybrid function.
- GA ignores the `ParetoFraction`, `DistanceMeasureFcn`, `InitialPenalty` and `PenaltyFactor` options.

The listed restrictions are mainly natural, not arbitrary. For example, there are no hybrid functions that support integer constraints, so GA does not use hybrid functions when there are integer constraints.

8.1 Genetic algorithm

In this chapter all issues related to the GA will be explained. That means detail all the parts of the code: the **fitness function**, and **configuration** of the GA (`gaoptimset`) as well as the methods developed in order to save time, improving the performance of the algorithm. First it will be presented some basic background of the genetic algorithms and its operation to be able to understand the following sections.

8.1.1 Overview of genetic algorithm

A genetic algorithm (GA) is a method for solving constrained and unconstrained optimization problems that based in **natural selection**, the process that drives biological evolution. The algorithm repeatedly modifies a population of individual solutions, selecting individuals at random from the current population to be parents and uses them to produce the children for the next generation. Along the successive generations, the population *evolves* towards an optimal solution, which is expected near a global minimum. The GA has been chosen for the optimization problem because it can address problems of **mixed integer programming** as this case studies.

8. Mixed integer optimization

To apply the natural selection process, the GA uses mainly three types of rules at each step to create the next generation from the current population:

- **Selection rules** select the individuals, called **parents**, which contribute to the population at the next generation.
- **Crossover rules** combine two parents to form children for the next generation.
- **Mutation rules** apply random changes to individual parents to form children.

To understand how this method differs from a classical algorithm, derivative-based, in the following table there are summarized the two main ways:

Classical algorithm	Genetic algorithm
Generates a simple point at each generation. The sequence of points approaches an optimal solution.	Generates a population of points at each iteration. The best point in the population approaches as optimal solution.
Selects the next point in the sequence by deterministic computation.	Selects the next population by computation which uses random number generators.

Table 8.1: Performance comparison classical algorithms and genetic algorithms.

To get a right understanding of the GA, it is needed some background with the most important concepts which a typical GA contains. Then the following list contains important definitions to be able to understand how the GA works:

- **Fitness function (fval)**: is the function to optimize, also called objective function.
- **Individuals**: any point to which fitness function will be applied. The value of the fitness function for an individual is its score. An individual is sometimes referred to as a **genome** and the vector entries of an individual as **genes**.
- **Populations and generations**; a population is an array of individuals. At each iteration, the GA performs a series of computations on the current population to produce a new population. A new generation is referred to each successive population.
- **Diversity**: refers to the average distance between individuals in a population. A population has high diversity if the average distance is large; otherwise it has a low diversity. Diversity is an important GA parameter, because it enables the algorithm to search a larger region of the space.

- **Fitness values** and **best fitness values**: the fitness value of an individual is the value of the fitness function for that individual; the best fitness value for a population corresponds to the smallest value for any individual in the population.
- **Parents** and **children**: to create the next generation, the GA selects certain individuals in the current population, called parents, and uses them to create individuals in the next generation, called children.

After defining the keywords about the GA, it is time to get an idea about how the algorithm works. The following outline summarizes how the GA performs:

1. The algorithm begins by creating a random initial population. Then creates a sequence of new populations. At each step, the algorithm uses the individuals in the current generation to create the next population. To create the new population, the algorithm performs the following steps:
 - a. Scores each member of the current population by computing its fitness value.
 - b. Scales the raw fitness scores to convert them into a more usable range of values.
 - c. Selects members, called **parents**, based on their fitness.
 - d. Some of the individuals in the current population that have lower fitness are chosen as **elite**. These individuals are passed to the next population.
 - e. Produces children from the parents. Children are produced either by making random changes to a single parent—**mutation**—or by combining the vector entries of a pair of parents—**crossover**.
 - f. Replaces the current population with the children to form the next generation.
2. The algorithm stops when one of the stopping criteria is met.

It is interesting to analyse step 2 more carefully because it is when genetic strategies take part into the algorithm. At each iteration, the GA uses the current population to create the children that make up the generation. The algorithm selects a group of individuals in the current population, parents, who contribute their genes—the entries of their vectors—to their children. The algorithm usually selects individuals that have better fitness values as parents. For the next generation the GA creates three types of children (figure 8.2):

- **Elite children** are the individuals in the current generation with the best fitness values. These individuals automatically survive to the next generation.

- **Crossover children** are created by combining the vectors of a pair of parents.
- **Mutation children** are created by introducing random changes, or mutations, to a single parent.

These are the most important strategies, but there is another interesting too, as for example **migration options**. This strategy specifies how individuals move between subpopulations. *Subpopulations* refer to a form a parallel processing for the GA, in which each worker hosts a number of individuals. These individuals are a subpopulation and the worker evolves the subpopulation independent from other workers, except when migration causes some individuals travel between workers. Migration occurs if the population size is set as a vector of length greater than 1, the best individuals from one subpopulation replace the worst individuals in another subpopulation. Individuals that migrate from one subpopulation to another are copied, they are not removed from the source subpopulation.

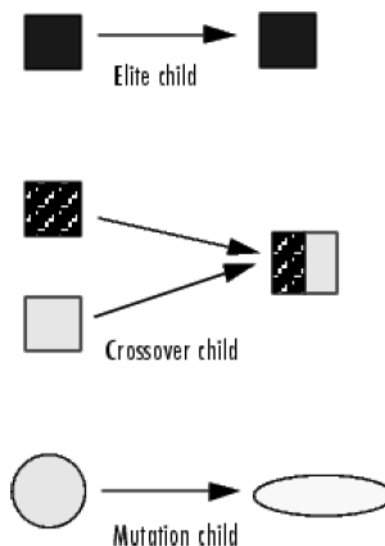


Figure 8.2: Schematic diagram which illustrates the three types of children. Extracted from [6].

In the following sections, all the parts of the algorithm will be exposed. First the fitness function, step by step how to obtain the objective value, then the options set for the GA, `gaoptimset`. In this second part it will also be presented the strategy developed to achieve the optimal solutions for each number of DVAs. Finally the last chapter is related to improving the performance of the algorithm: **vectorized code**, **parallel processing** and **refinement** of the GA. This last chapter is the most important one in terms of achieving one of the requirements set in this study: to develop an optimization algorithm computationally cheap, which means saving computing time.

8.1.2 Fitness function

The fitness function is the part of the algorithm which calculates the value to minimize, **fval**. This function is clue to be able to speed-up the code that is because it is repeated iteratively for each individual of the population in each generation. By average the GA normally, converges in values near to 20 generations, that means 2000 individuals, so the fitness function is run 2000 times. So it is clearly that running the fitness function takes most of the computing time, which is why most of the efforts of this study are related about trying to speed up this function (see chapter 8.1.4).

This function, as it has been explained briefly in previous chapters, contains two important functions: **DVAParameters.m** and **H_UnitaryExternalForce_E1.m** (these functions have been supplied by the LEAM). Furthermore taking the outputs of **H_UnitaryExternalForce_E1.m** it is possible to, after some calculation and applying the normative, obtain the **fval**. The process step by step is:

1. To generate the *struct* **DVAs**, by calling the function **DVAParameters.m**. This *struct* contains all the parameters of the DVAs.
2. To call **H_UnitaryExternalForce_E1.m**; the output of this function is the receptance of unitary external force at evaluator on surface in the presence of DVAs.
3. To apply the normative (ISO 2631-2) and final calculations to obtain the objective optimization value, **fval**.

Step three uses the output matrix from step two to obtain the **power spectral density** (PSD), which is the minimization objective value. Anyway, in following subsections will be explained clearly each of these steps as well as inputs and outputs which are needed.

As well as in the case of the GA, the fitness function has been modified in for each optimization case. So it has been implemented in total, ten different fitness functions corresponding to each number of DVAs. In fact changes are slightly because just requires to introduce the new known from the previous DVAs calculation to the **DVAParameters.m** function, and the rest of function remains without changes. All the changes and the different versions of the fitness function and the GA will be explained and justified in detail in chapter 8.1.4.4. In any case, in annex A (subsection A.1), different versions of the code are presented, each one corresponding to a different case in terms of number of DVAs.

8.1.2.1 Step 1: Generate DVASET

This first step just run DVAParameter.m by using as inputs the arrays supplied by the GA as individuals and generates the *struct* **DVASET**, which is one of the inputs for the next step. This *struct* includes the following information:

- **NodeID**: express the **position** where DVA is applied in the cross section of the tunnel. This aspect is related to one of the other inputs of the step two, which will be explained later. The possible nodes go from 1 to 39 and of course one DVA is set in a position, this node is unavailable for the rest of DVAs.
- **Distribution**: this parameter can be '**Periodic**' or '**Discrete**'. In this study only the periodic distribution case has been used.
- **DistributionProperties**: defines the distances between longitudinal DVA distributions in meters (that means x-axis according to the coordinates system used in this study).
- **Numberof_LD_DVA**: is the number of longitudinal DVA for each type of DVA associated at each node. This means that each DVA set in a node will be repeatedly placed along the longitudinal direction according to this value.
- **k, m, c**: mechanical properties of the DVAs; **stiffness** ($\text{N}\cdot\text{m}^{-1}$), **mass** (kg) and **damping** of the absorbers ($\text{N}\cdot\text{s}\cdot\text{m}^{-1}$), respectively.

An example of how to call correctly this function, first prepare the values of the parameters (table 8.2):

Parameters	Units	Values
NodeID	-	{32; 8}
DistributionProperties	m	{1; 1}
Numberof_LD_DVA	-	{9; 9}
k	$\text{N}\cdot\text{m}^{-1}$	{4.5e+07; 4.6e+07}
c	$\text{N}\cdot\text{s}\cdot\text{m}^{-1}$	{4.5e+05; 0.01*4.6e+05}
m	kg	{950; 960}

Table 8.2: Example of possible parameters used to generate DVASET.

8. Mixed integer optimization

Table 8.2 presents the parameters of the DVAs, but for example for this study case, as it is known that generally $C = 0.01 \times K$, then damping of the absorbers is not a variable in the GA because it is obtained from K . Finally it is time to call the function, as:

- ```
DVASET = DVAParameters({32;8},{ 'Periodic';
 'Periodic'}, {1;1}, {9;9}, {4.5e+07;4.6e+07},
 {4.5e+05;0.01*4.6e+05}, {950; 960});
```

This example is quite interesting because it uses more than one DVA so it requires the use of '{' and '}' to entry the values of both DVAs in each parameter of the *struct*. The *struct* DVASET will be, then, an input for H\_UnitaryExternalForce\_E1 and all the values will be extracted from it by using the standard syntax of MATLAB (see reference [28]). In annex A (subsection A.2) there is a schema which represents this DVASET in the tunnel.

### 8.1.2.2 Step 2: Call H\_UnitaryExternalForce\_E1.m

This second part, mainly develops the theory exposed in chapter 7, to obtain the receptance of unitary external force at evaluator on surface in the presence of DVAs. This function is, for sure, the most complex of the algorithm. This input has two inputs, at one side there is the *struct* DVASET (obtained previously from step 1) and at the other side the archive **H\_DVA+Eval\_SpaceDomain.mat**. This archive consists of information as:

- fs**: is the range of **frequencies** used in the study, from 1 to 100 Hz.
- Evaluators**: is an 80x8 dataset; a table which includes important information to placed DVAs and to study the vibration response. It contains information about all the possible NodeID values as well as its coordinates (**xNode** and **yNode**), the same using **ForceID**, and characterizing it by **ForceType** (can be **'Ext'** or **'Int'**). It also sets the vector components of the force, by **xForce** and **yForce** and if this force is 3D or not (**ThreeD\_Normal**). To see in detail this data and the range of values in which it works, see annex B.
- H\_DVA**: is 100x1024x160 matrix (complex double type), which shows the receptance of unitary external force at position of external force and all possible DVA position. Furthermore, the receptance of unitary external force and unitary force in each DVA position at a final evaluator on surface.
- kz**: is presenting **wavenumber** sampling.

With these two inputs the function manages to obtain a matrix. This matrix is the receptance of the system and is a 3D matrix; gives the information in the form  $[N_w, N_z, NDOF]$ . Where  $N_w$  is frequency (from 1 to 100 as  $fs$ ; see step 1),  $N_z$  is longitudinal distance (displacement) and  $NDOF$  refers to the degrees of freedom which are  $x, y$  and  $z$ , according to the coordinate system, so the matrix will be of the form  $100 \times N \times 3$ , first dimension corresponds the range of frequency, the second depends on the number of DVAs placed and the third to the  $NDOF$ , in this study  $[x, y, z]$ .

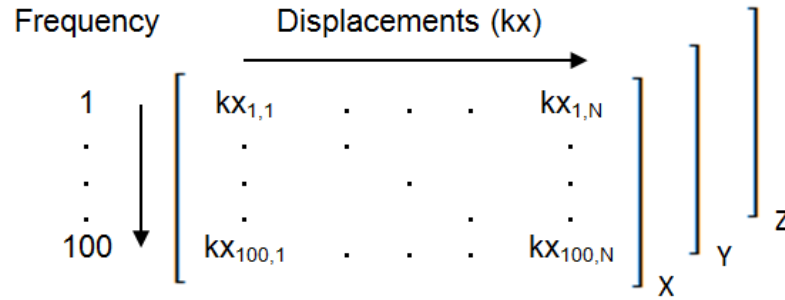


Figure 8.3: Example of the output of H\_UnitaryExternalForce\_E1.m.

In figure 8.3, there is an example of the matrix. It contains all the displacements values, where rows are associated to the corresponding value of frequency and for each degree of freedom, in this study  $x, y$  and  $z$ .

About using this function, it is also important to know that to be able to run it, are required two other external functions, which are used to do some complex operations and help to save time. Both functions are available free, in a file exchange platform. These two functions are:

- **mtimesx.m**: fast matrix multiplier with multi-dimensional support [29].
- **MultiSolver.m**: multiple linear (least-square) solver for systems having the same size [30].

After obtaining the resulting matrix, it is time to prepare results to be able to apply the normative in step 3. The matrix is computed for each dimension and frequency values using  $\ddot{X} = -\omega^2 \cdot X$ , where  $\ddot{X}$  is the acceleration in  $\frac{m}{s^2}$ ,  $\omega$  is the frequency in hertz and  $X$  is the displacement (kx) in meters. Then all the values of each row are added by a simple sum and finally it is applied a sum of squares to each dimension value as  $\left( \sqrt{|\ddot{X}|^2 + |\ddot{Y}|^2 + |\ddot{Z}|^2} \right)^2$ , where  $\ddot{X}, \ddot{Y}$  and  $\ddot{Z}$  are the acceleration values for each dimension. For better understanding of the function see annex A (subsection A.3).

### 8.1.2.3 Step 3: ISO – 2631-2 Application

This final step inside the fitness function is in which normative ISO 2631-2 is applied. This normative evaluates human exposure to whole-body vibration. Structural vibration to which human beings are exposed in buildings can be detected by the occupants can affect them in many ways. More particularly, their comfort and quality of life may be reduced. It specifies a method for measurement and evaluation of the range of frequencies which affect in this way to the comfort of the occupants. Furthermore It defines the frequency weighting  $W_m$  which is applicable in the frequency range 1 Hz to 80 Hz and extended to 100 Hz [31].

In the mathematical approach frequencies  $f_i$  ( $i = 1$  to 3) are parameters of the transfer function determining to overall **frequency weighting**  $W_m$ . The transfer function,  $H(p)$ , is expressed as the product of three factors, **high-pass filter**  $H_h(p)$ , **low-pass filter**  $H_l(p)$  and **pure weighting function**  $H_t(p)$ , as follows, where  $\omega_i = 2\pi f_i$  and  $p = j2\pi f$ :

Band limiting (filters with second-order Butterworth characteristic;  $f_1$  and  $f_2$  are the corner frequencies) and the pure frequency weighting are shown in table 8.3.

| High pass                                                      | Low pass                                                       | Pure frequency weighting                                          |
|----------------------------------------------------------------|----------------------------------------------------------------|-------------------------------------------------------------------|
| $H_h(p) = \frac{1}{1+\sqrt{2}\omega_1/p+(\omega_1/p)^2}$ (8.1) | $H_l(p) = \frac{1}{1+\sqrt{2}p/\omega_2+(p/\omega_2)^2}$ (8.3) | $H_t(p) = \frac{1}{1+p/\omega_3}$ (8.5)                           |
| $ H_h(p)  = \sqrt{\frac{f^4}{f^4+f_1^4}}$ (8.2)                | $ H_l(p)  = \sqrt{\frac{f_2^4}{f^4+f_2^4}}$ (8.4)              | $ H_t(p)  = \sqrt{\frac{f_3^4}{f^4+f_3^4}}$ (8.6)                 |
| $f_1 = 10^{-0.1}\text{Hz} = 0.7943 \text{ Hz}$                 | $f_2 = 100 \text{ Hz}$                                         | $f_3 = \frac{1}{0.028 \times 2\pi} \text{ Hz} = 5.684 \text{ Hz}$ |

Table 8.3: Formulas for generating the transfer function with the corresponding values of  $f_1$ ,  $f_2$  and  $f_3$ .

The transfer function  $H(p)$ , of the band-limited frequency weighting  $W_m$  is given by the product of the high-pass filter  $H_h(p)$ , the low-pass filter  $H_l(p)$  and the pure weighting function  $H_t(p)$ :

$$H(p) = H_h(p) \cdot H_l(p) \cdot H_t(p) \quad (8.7)$$

Finally just calculate **fval**, which will be the **power spectral density** (PSD) in  $\left(\frac{m}{s^2}\right)^2$  / Hz :

$$\text{PSD} = \frac{1}{N^2} \sum_{m=1}^N (|U_m^2(\omega)|^2 \cdot |H_m(\omega)|^2) \quad (8.8)$$

Where  $N$  is the number of points. As the range of frequency is from 1 to 100, then  $N = 100$ .  $H_m(\omega)$  corresponds to the values obtained before,  $H(p)$  for each frequency and  $U_m^2$  with units  $\left(\frac{m}{s^2}\right)^2$ , it has been calculated in the step 2 as:

$$U^2 = \left( \sqrt{|\ddot{X}|^2 + |\ddot{Y}|^2 + |\ddot{Z}|^2} \right)^2 \quad (8.9)$$

### 8.1.3 Configuration of the GA

Configuration of the GA is a key point in order to achieve good performance of the algorithm. This is the configuration of the GA for the case of one DVA:

```
%BOUNDS

%lower bound
lb = [1 1 2 0.5*10^7 500];
%upper bound
ub = [39 10 40 0.5*10^8 1000];

%options
opts = gaoptimset(...
 'Display','iter', ...
 'CrossoverFraction', 0.6, ...
 'EliteCount', 10, ...
 'Generations', 300, ...
 'MigrationDirection', 'both', ...
 'MigrationInterval', 10, ...
 'MigrationFraction', 0.2, ...
 'PlotFcns', {@gaplotbestf,@gaplotbestindiv, ...
 @gaplotdistance,@gaplotgenealogy}, ...
 'PopulationSize', 100, ...
 'PopulationType', 'doubleVector', ...
 'StallGenLimit', 15, ...
 'StallTimeLimit', 10800, ...
 'TolFun', 1e-10, ...
 'UseParallel', true, ...
 'Vectorized', 'off');

rng(1, 'twister')
[x fval exitflag] =
ga(@fitness_1DVA,5,[],[],[],[],lb,ub,[],[1 2 3],opts);
```



Despite of not all the GAs have been configured in the same way (there are slight differences for each number of DVAs), they have the same structure:

1. Set up the **bounds**: this part set the maximum and minimum values of the inputs. The GA generates a vector containing all the variables, see table 8.4:

| Vector component | Variable               |
|------------------|------------------------|
| $x(1)$           | NodeID                 |
| $x(2)$           | DistributionProperties |
| $x(3)$           | Numberof_LD_DVA        |
| $x(4)$           | Stiffness (k)          |
| $x(5)$           | Mass (m)               |

Table 8.4: Inputs of the GA.

As it has been explained in chapter 8.1.2 Fitness function, Distribution and C (damping of the absorbers) are not set as input, so the problem has five unknown variables. Bounds have been fixed according to different criteria, for example K and M because of the mechanical limitations of the DVAs and DistributionProperties and Numberof\_LD\_DVA due to the result of simulations show that the first one is always near to one and the other fluctuates between eight and twenty, so it has also been included some margin. Finally NodeID is set by the Evaluators, see annex B.

2. Set up the **options**: options are configured using `gaoptimset` function. The configuration settings are limited by be a mixed integer problem; the most important are shown in 8.5 and in annex A (subsection A.5) there is an example of 'PlotFcns'.
3. Calling the GA: fitness function, bounds and option are introduced in the algorithm. `[1 2 3]` defines the variables of the vector which are integers ( $x(1)$ ,  $x(2)$  and  $x(3)$ ) and the rest of empty brackets are because there are no constraints. The function `rng(1, 'twister')` is useful to manage the random numbers generated during the iterations.

Finally some justifications about the configurations settings as well as the strategy developed to the cases for more than one DVA will be exposed in section 8.1.4.4 Refinement of the GA.

| Configuration settings                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 'Display', 'iter'                                                                           | Information of the GA is displayed in each iteration.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 'CrossoverFraction'                                                                         | Specifies the fraction of the next generation, other than elite children, that are produced in crossover; has to be a fraction between 0 and 1.                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 'EliteCount'                                                                                | Specifies the number of individuals that are guaranteed to survive to the next generation. It has to be a positive integer less than or equal to the population size.                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 'Generations'                                                                               | Specifies the maximum number of iterations for the GA to perform.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 'MigrationDirection',<br>'both'                                                             | Specifies that $n$ subpopulation migrates into the $(n-1)$ and the $(n+1)$ subpopulation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 'MigrationInterval'                                                                         | Specifies how many generations pass between migrations.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 'MigrationFraction'                                                                         | Specifies how many individuals move between subpopulations                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 'PlotFcns',<br>{@gaplotbestf,<br>@gaplotbestindiv,<br>@gaplotdistance,<br>@gaplotgenealogy} | <ul style="list-style-type: none"> <li>• @gaplotbestf: plots the best function value versus generation.</li> <li>• @gaplotbestindiv: plots the vector entries of the individual with the best fitness function value in each generation.</li> <li>• @gaplotdistance: plots the average distance between individuals at each generation.</li> <li>• @gaplotgenealogy: plots the genealogy of individuals. Lines from one generation to the next are color-coded, red lines indicate mutation children, blue lines crossover children and black lines indicate elite individuals.</li> </ul> |
| 'PopulationSize'                                                                            | Specifies how many individuals there are in each generation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 'PopulationType',<br>'doubleVector'                                                         | Specifies the type of input to the fitness function. 'doubleVector' is used in case of mixed integer programming.                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 'StallGenLimit'                                                                             | The algorithm stops if the average relative change in the best fitness function value over stall generations is less than or equal to function tolerance.                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 'StallTimeLimit'                                                                            | The algorithm stops if there is no improvement in the best fitness value for an interval of time in seconds specified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 'TolFun'                                                                                    | The algorithm stops if the average relative change in the best fitness function value over stall generations is less than or equal to function tolerance.                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 'UseParallel', true                                                                         | GA calls the fitness function in parallel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 'Vectorized', 'off'                                                                         | It is mandatory in order to use parallel processing,                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

Table 8.5: Configuration options of the GA.

## 8.1.4 Performance improving methods

In following sections will be presented methods which have been implemented in order to speed up the algorithm, **vectorization of the code** and **parallel processing** are related to computing the fitness function, while the **refinement of the GA** works in how to change the parameters of the GA to improve performance of the algorithm.

### 8.1.4.1 Vectorization of the code

The first version of the fitness function was using `for` structures, using loops so most of the time was spent into indexing matrices and arrays. In this way with the aim of saving time, it has been applied a **vectorization** of the fitness function. Vectorization is the process of revising loop-based, scalar-oriented code to matrix and vector operations. This method is worthwhile for several reasons:

- **Appearance:** vectorized mathematical code appears more like the mathematical expressions found in textbooks, making the code easier to understand (see table 8.6).
- **Less error prone:** without loops, vectorized code is often shorter. Fewer lines of code mean fewer opportunities to introduce programming errors.
- **Performance:** vectorized code often runs much faster than the corresponding code containing loops.

| Initial code                                                                                                                                                                                                                                                                                                                                 | Vectorized code                                                                                                                                                                                                                           |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>% ISO NORMATIVE  f1 = 10^-0.1;          % [Hz] f2 = 100;              % [Hz] f3 = 1/(0.028*2*pi);   % [Hz]  for i = 1:length(fs)     H_h(1,i) =     sqrt(fs(1,i)^4/(fs(1,i)^4+f1^4));     H_l(1,i) =     sqrt(f2^4/(fs(1,i)^4+f2^4);     H_t(1,i) =     sqrt(f3^2/(f3^2+fs(1,i)^2));     H(1,i) =     H_h(1,i)*H_l(1,i)*H_t(1,i);</pre> | <pre>% ISO NORMATIVE  f1 = 10^-0.1;          % [Hz] f2 = 100;              % [Hz] f3 = 1/(0.028*2*pi);   % [Hz]  H_h = sqrt(fs.^4./(fs.^4+f1^4)); H_l = sqrt(f2^4./(fs.^4+f2^4); H_t = sqrt(f3^2./(f3^2+fs.^2)); H = H_h.*H_l.*H_t;</pre> |

Table 8.6: Comparative of the initial version of the code and the vectorized form with an extract of the code in which the normative is applied.

### 8.1.4.2 Parallel processing

Parallel processing is an interesting way to speed the optimization algorithm. This feature has been developed using a MATLAB application, so it is needed to have a **Parallel Computing Toolbox license**, and have configured a parallel worker pool. The idea is to take profit of the possibility of speed-up the GA by evaluating population in parallel. This option is incompatible with vectorization of fitness function or set up some constraint functions.

An inconvenient related to using parallel processing in MATLAB is that random numbers sequences in MATLAB are pseudorandom, determined from a seed, or an initial setting. Parallel computations use seeds that are not necessarily controllable or reproducible, according to this, for the specific case of GA parallel population generation gives **non-reproducible results**.

Function GA can automatically distribute the evaluation of objective and nonlinear constraint functions associated with a population to multiple processors. To be able to implement GA using parallel computing is needed to:

- Have a license for **Parallel Computing Toolbox** software.
- Enable parallel computing with `parpool`, a Parallel Computing Toolbox function.
- Set the following options using `gaoptimset`:
  - '**Vectorized**' is '**off**'; this option determines whether GA evaluates an entire population with one function call in a vectorized fashion.
  - '**UseParallel**' is **true**; it has to be selected in order to be able to compute the fitness function in parallel.

When these conditions hold, GA is able to compute the fitness function of the individuals in a population in parallel. Even the algorithm is running in parallel, GA occasionally calls the fitness and nonlinear constraint functions serially on the host machine. To compute the fitness function of the individuals in parallel means an advantage in terms of saving time respect computing it in series. The other option is to set in '**on**' the option **Vectorized**, that means shut down also parallel computing. But it has been tested for this case study that is not a good option because of how the fitness function works. This configuration works better in cases where the biggest computing charge is in vector form, but in this case most of the computing is in the function `H_UnitaryExternalForce_E1.m`.

### 8.1.4.3 Study of parallel pool configuration

Using a computer with a **multicore processor** is possible to speed up the algorithm using parallel processing. The objective is to establish a parallel pool of several workers with Parallel Computing Toolbox license. The main function to generate these workers is **parpool**, the elemental ways to use this function are:

- **parpool**: entering this code at the command line, MATLAB starts a pool of workers using the multicore processor.
- **parpool('local')**: this function is used in case a nondefault cluster profile has been set, so it is possible to enforce multicore (local) computing.

This function enables the full functionality of the parallel language features in MATLAB by creating a special job on a pool of workers, and connecting the MATLAB client to the parallel pool, then are listed all the possible syntaxes options:

1. `parpool`
2. `parpool(poolsize)`
3. `parpool(profilename)`
4. `parpool(profilename,poolsize)`
5. `parpool(cluster)`
6. `parpool(cluster,poolsize)`
7. `parpool(___,Name,Value)`
8. `poolobj = parpool(___)`

In this study option four has been implemented. With this syntax it is possible to set up the number of workers using the default cluster profile (`profilename='local'`). The other option, **poolsize**, overrides the number of workers specified in the preferences or profile, and starts a pool of exactly that number of workers, even if it has to wait for them to be available. Finally to shut down parallel pool is used **delete(gcp)**.

In order to find the optimal number of workers for the computing machine in which the optimization process has been developed, it was necessary to design a practical evaluation of processing times for each number of workers. Then with this aim the code has been slightly modified to attend to these constraints. The objective is to execute the GA with a reduction of **PopulationSize** and **Generations** values, few times for each number of workers, to be able to evaluate mainly, processing time.

First, it is important to understand what a **parallel pool** is:

- A parallel pool is a set of **workers** in a **compute cluster** or desktop, bound together by a special type of job so they can be used interactively and can communicate with each other during the lifetime of the job. A parallel pool normally describes this collection of workers, such as when referring to the pool size. While these pool workers are reserved for your interactive use they are not available to other uses [7].

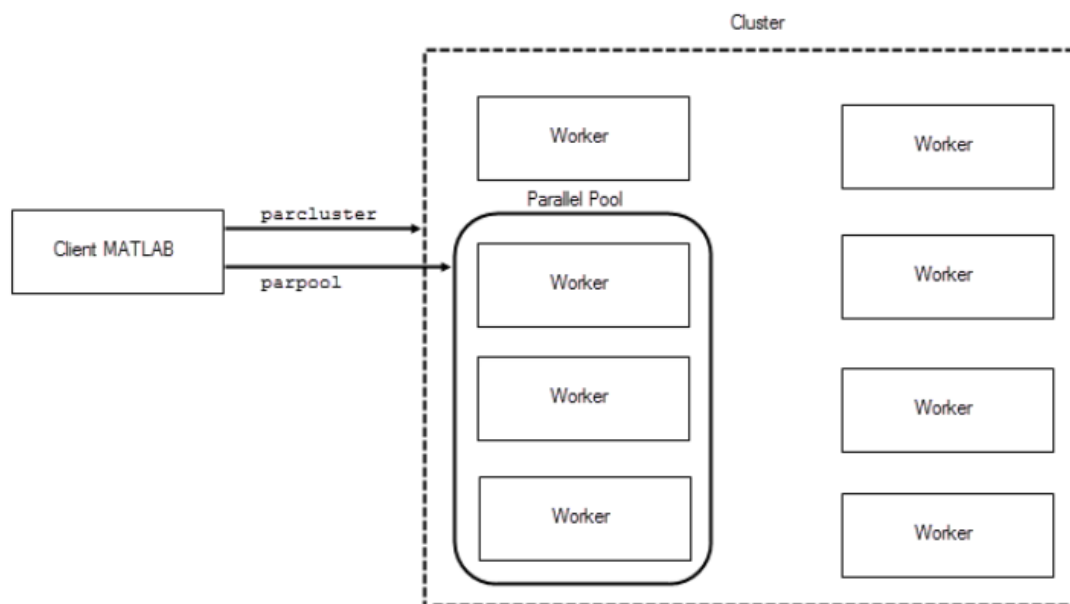


Figure 8.4: Schematic view of how parallel pool works. It is only possible to have one parallel pool accessible at time from a MATLAB client session. Image from [7].

To set up the experiment it is important to clearly identify the important variables; in this case there are two:

- **Computing time:** is the most important variable, because this study has as an objective to generate a code to be able to find a solution computationally cheap. In terms of time, that means that reducing computing time is considered as priority.
- **Performance the computer:** the other important variable is performance of the computer. In this case, it is related to the **temperatures of the cores** (multi-core processor) and **CPU usage** as well as **physical memory usage**.

Computing time has been evaluated using `tic` and `toc` MATLAB functions. CPU and physical memory usage have been measured using simply **Task Manager** included by default in Windows operating system. Finally the temperature of the cores has been

evaluated using **CPUID HWMonitor**, it is a hardware monitoring program that reads PC systems main health sensors: voltages, temperatures and fans speed [32].

To be able to compare the results, the conditions of each case study have to be the same. According to this the parameters of the GA have been set for all the cases and iterations to:

- 'PopulationType', 'doublevector'.
- 'PopulationSize', 20.
- 'Generations', 1.
- 'EliteCount', 1.
- 'UseParallel', true.
- 'Vectorized','off'.

These two options have been reduced from the initial GA because it is not needed to run the complete GA just to evaluate the processing time. Just knowing how much time spends for one generation it is possible to compare the values for each case. The idea is to get easily results then it is enough with one generation to analyse how parallel pools are working.

The code has been implemented using a `for` loop from 1 to 20 as the number of workers. The upper bound of workers has been set in twenty because for this parallel pool configuration, temperatures reached in the cores increased near to 100°C, which is fixed as the  $T_{JUNC}$  by the maker (**junction temperature** is the maximum temperature allowed at the processor die [33]). The time values have been stored in a matrix, as for each parallel pool configuration has been done five iterations the final matrix has 20x5 dimensions.

**Task manager** and **CPUID HWMonitor** have been running in parallel, monitoring the performance of the computer. Increasing the number of workers, multi-cores start to get hotter until they reach a bound of optimal temperature in a maximum of 96°C. About the physical memory, results have been quite predictable. As the performance of a multi-core processor is limited by the number of cores, in this case it has been used a **4 dual-core** so the maximum number of workers without generate more workers than cores are 8. So for this value it is expected to have a top performance respect the average. From 8 up to 20 because of the restriction of number of cores, MATLAB internally separated the cores to obtain the defined value. This operation does not guarantee a good performance because it generates overcharge the other important parameters. In case of physical memory and CPU usage, up to 8 workers the requirements are so high that it is difficult for the computer to run other tasks in parallel.

| Workers | Time 1 [s] | Time 2 [s] | Time 3 [s] | Time 4 [s] | Time 5 [s] |
|---------|------------|------------|------------|------------|------------|
| 1       | 736.32     | 762.86     | 739.32     | 718.02     | 701.95     |
| 2       | 413.15     | 403.98     | 408.43     | 422.36     | 406.02     |
| 3       | 287.49     | 285.17     | 285.44     | 283.60     | 284.11     |
| 4       | 239.41     | 239.90     | 240.72     | 242.96     | 241.56     |
| 5       | 225.14     | 220.65     | 222.35     | 224.40     | 222.62     |
| 6       | 199.27     | 200.54     | 199.33     | 196.06     | 199.15     |
| 7       | 190.48     | 191.11     | 192.74     | 190.40     | 190.95     |
| 8       | 186.84     | 183.72     | 184.66     | 181.83     | 182.54     |
| 9       | 181.50     | 182.05     | 193.94     | 182.81     | 189.24     |
| 10      | 192.40     | 202.66     | 183.40     | 182.63     | 183.11     |
| 11      | 183.54     | 182.73     | 181.84     | 180.68     | 183.38     |
| 12      | 186.50     | 183.00     | 183.96     | 205.44     | 193.35     |
| 13      | 200.38     | 200.71     | 206.88     | 197.90     | 211.12     |
| 14      | 196.89     | 191.49     | 192.88     | 211.50     | 213.31     |
| 15      | 211.80     | 218.11     | 210.03     | 207.20     | 212.66     |
| 16      | 223.58     | 206.53     | 226.90     | 218.95     | 213.12     |
| 17      | 228.39     | 229.65     | 230.48     | 227.43     | 224.48     |
| 18      | 254.06     | 233.35     | 215.20     | 226.53     | 223.35     |
| 19      | 223.67     | 213.41     | 217.62     | 212.25     | 213.25     |
| 20      | 210.70     | 209.47     | 210.97     | 210.92     | 211.47     |

Table 8.7: Computing time obtained for each parallel pool configuration and iterations.

In table 8.7 there are all the values stored in the matrix. To better understand these results, **best** and **mean values** have been selected. In this way it is possible to have a better idea of how the different parallel pool configuration changes the performance of the computer. From table 8.8, as it was expected, using 8 workers shows one of the best time results. In this case the optimal point is a pool of 11 workers and from this value, times start to increase, probably because of temperatures also increase, dangerously and CPU and physical memory started to collapse.

So finally, according to the time results shown in table 8.8, and the performance of the computer in each case, the final value selected for the parallel pool configuration has been 8 workers. This solution represents the optimal solution in terms of time efficiency and of course being conservative with the performance of the computer.



| Workers | Best [s] | Mean [s] | Average [s] |
|---------|----------|----------|-------------|
| 1       | 701.95   | 762.86   | 731.70      |
| 2       | 403.98   | 422.36   | 410.79      |
| 3       | 283.60   | 287.49   | 285.16      |
| 4       | 239.41   | 242.96   | 240.91      |
| 5       | 220.65   | 225.14   | 223.03      |
| 6       | 196.06   | 200.54   | 198.87      |
| 7       | 190.40   | 192.74   | 191.14      |
| 8       | 181.83   | 186.84   | 183.92      |
| 9       | 181.50   | 193.94   | 185.91      |
| 10      | 182.63   | 202.66   | 188.84      |
| 11      | 180.68   | 183.54   | 182.44      |
| 12      | 183.00   | 205.44   | 190.45      |
| 13      | 197.90   | 211.12   | 203.40      |
| 14      | 191.50   | 213.31   | 201.21      |
| 15      | 207.20   | 218.11   | 211.96      |
| 16      | 206.53   | 226.90   | 217.82      |
| 17      | 224.48   | 230.48   | 228.08      |
| 18      | 215.20   | 254.06   | 230.50      |
| 19      | 212.25   | 223.67   | 216.01      |
| 20      | 209.47   | 211.47   | 210.70      |

Table 8.8: Best time values, mean time values and averages values. In green the two best options.

#### 8.1.4.4 Refinement of the GA

In this chapter it will be presented the configurations set in the algorithm as a part of the refinement process of the GA. There are two main points in order to improve the performance of the algorithm, modifying the GA parameters to speed up the convergence and set up the strategy to the optimal cases for more than one DVA. Then both points are explained:

- **GA parameters:** the important parameters to be refined are the ones related to the genetic strategies; **crossover**, **migration**, **mutation**, **elite children** ... In this way it has been set these parameters as it is possible to see in table 8.9. The objective of changing these parameters is to obtain more diversity in the individuals along the populations to reach faster the optimal solution. For example it is important to set up the population size according of the specifications of the problem. With a large population size,

the GA searches the solution space more thoroughly, thereby reducing the chance that the algorithm returns a local minimum that is not a global minimum. However, a large population size also causes the GA to run more slowly, this is why by increasing the number of DVAs included in the problem, the population size has been reduced and parameters as mutation and migration have been increased to generate more diversity in lowest range of individuals.

| GA parameters       | Value      |
|---------------------|------------|
| 'CrossoverFraction' | 0.6        |
| 'EliteCount'        | 10         |
| 'Generations'       | 300        |
| 'MigrationInterval' | 10         |
| 'MigrationFraction' | 0.2        |
| 'PopulationSize'    | 150        |
| 'StallGenLimit'     | 15         |
| 'StallTimeLimit'    | 10,800 [s] |
| 'TolFun'            | 1e-10      |

Table 8.9: GA parameters for the optimization of the case of two DVAs.

`CrossoverFraction` also allows configuring `MutationFraction`, because they are complementary parameters. This means that for the `CrossoverFraction` value of table 8.1, will be 0.4, because there the two only ways of generating children apart from the elite children. These parameters are probably the most important ones; with low values of mutation, the GA cannot create enough new genes, so in case that the optimal point has not been reached yet, it will be difficult that the GA will find it. In the other way, if mutation level is too high, the random changes that the GA applies will difficult improve the fitness value and as crossover is low not many of these changes will be saved. Then the objective is to find an equilibrium value in which the GA gives the best performance.

As it has been explained before, these parameters have to be adapted to the case that is being studied. For example population size has decreased if computing time of the fitness function grows. This is how it has been done in this study for the different cases, adapting these settings, slightly, to obtain better performances for each number of DVAs.

- Optimization strategy:** this strategy is an important issue to solve the cases for more than one DVA. There is no problem for just one DVA, but when trying to solve the optimization problem for two DVAs computing time increases too much as for considering the algorithm is performing enough faster. Because of that it has been considered the need to design a strategy to improve the performance of the algorithm. In this way, it has been investigated and tested the code for more multiple until a solution is found: optimal solution for  $n$  DVAs is equal to the solution for  $n - 1$  DVAs, plus the result of GA run for the  $n$  DVAs but fixing the values of  $n - 1$  DVAs. To see it clearly there is an example of the strategy proposed in table 8.10, where is solved the optimization of the case of two DVAs. About this strategy, it is important to understand that the advantage is that does not need to calculate each DVA every time; for each new number of DVAs it is just needed to compute one DVA, using he previous ones.

| Optimal solution for 2 DVAs                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                 |                        |            |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|-----------------|------------------------|------------|
| 1. Find the optimal solution for 1 DVA                                                                                                                                                                                                                                                                                                                                                                                                                                        |                        |                 |                        |            |
| NodeID                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | DistributionProperties | Numberof_LD_DVA | k [N·m <sup>-1</sup> ] | m [Kg]     |
| 32                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 1                      | 9               | 4.9998e+07             | 999.9947   |
| 2. Use this value as a part of the optimal solution for 2 DVAs, so this solution has to be added in the fitness function for two DVAs as: <pre> NodeID = {32; x(1)}; DistributionProperties = {1; x(2)}; Numberof_LD_DVA = {9; x(3)}; K = {4.9998e+07; x(4)}; C = {0.01*4.9998e+07; 0.01* x(4)}; M = {999.99; x(5)};           </pre> Where $x(1)$ , $x(2)$ , $x(3)$ and $x(4)$ are the variables set as input for the GA and they will define the parameters of the new DVA. |                        |                 |                        |            |
| 3. The final solutions for two DVAs are the optimal solution for 1 DVA plus the result obtained from the GA.                                                                                                                                                                                                                                                                                                                                                                  |                        |                 |                        |            |
| NodeID                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | DistributionProperties | Numberof_LD_DVA | k [N·m <sup>-1</sup> ] | m [Kg]     |
| 32                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 1                      | 9               | 4.9998e+07             | 999.99     |
| 8                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 1                      | 9               | 4.9999e+07             | 4.9999e+05 |

Table 8.10: Example of the strategy followed by the algorithm for the case of two DVAs.

The objective of the refinement process is to save time without decreasing the accuracy of the algorithm. The strategy proposed has been fundamental in order to obtain results presented in chapter 9. Results and analysis, it has become possible to be able to obtain optimal solution for more than one DVA, in a proper amount of time. And of course, as it has been explained in previous subsections, there is a problem in mixed integer optimization related to constraints, this strategy allows avoiding this problem. Furthermore, fitness function has been modified in order not to put a new DVA in a filled `NodeID`. It has been used a vector with all the possible positions in which a new DVA can be placed:

```
Pop_val = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
34 35 36 37 38 39];

%excluded NodeID 32

x(1) = Pop_val(x(1));
```

In this way, now the first input of the GA refers to the index of the vector containing the possible positions. This fragment of the code is part of the fitness function for 2 DVAs. Position 32 has been excluded because it corresponds to the optimal solution for one DVA, so in this position it cannot be placed a new DVA.

### 8.1.5 Improving accuracy using pattern search

In these sections, it has been proposed a model to be able to release the computing charge of the GA by using a **pattern search** algorithm. As it has been explained in chapter 8, the GA configuration does not accept to set up a hybrid algorithm in case of mixed integer programming, which is why it has to be done manually, by selecting the variables of type **double**. The idea is to run the GA and when it is finished, take the variables of type double, **stiffness** and **mass**, and run to pattern search to find more accurate values of these variables.

This model has been designed to be used for cases in which computing time of the GA is considerably high. With the computer in which this study has been developed, from the optimization for six DVAs, computing time of the GA starts to grow very quickly. Most of this time is because of the variables of type double, as the other variables are integers the optimal value for these ones is faster to be obtained, but the GA needs more time to reach the optimal values for the double one, furthermore the tolerances specifications (presented in section 8.1.3 Configuration of the GA) have to be

accomplished. So the objective of combining this algorithm with the GA generating a hybrid algorithm is to be able to find accurate values of stiffness and mass saving time.

About the solver selected, pattern search (PS) is a numerical optimization method that does not require the gradient of the problem to be optimized. Then the PS can be used to optimize functions that are **not continuous** or **differentiable**. This solver has been selected because it is possible to easily adapt the fitness function, other solvers needed a deep treatment of the fitness function. Furthermore, despite of PS is slower than gradient-base solvers, it is able to run in parallel, so it becomes faster.

This is an example of how to implement the pattern search; this case is for one DVA (to see the complete algorithm see annex A, subsection A.5):

```
%BOUNDS
%lower bound
lb = [0.5*10^7 500];
%upper bound
ub = [0.5*10^8 1000];

x0 = [x_ga(4) x_ga(5)];

options = psoptimset(...
'Tolfun', 1e-10, ...
'TolX', 1e-10 ...
);

x_PS = patternsearch(@PS_1DVA,x0,[],[], ...
[],[],lb,ub,[],options);
```

The **bounds** work in the same way as the GA. In this solver is needed an **initial point**,  $x_0$ , which has been set as the values obtained from the GA ( $x_{ga}(4)$  and  $x_{ga}(5)$ ) corresponding to the stiffness and mass. The fitness function is slightly the same as in the GA but just taking as variables the stiffness and mass, the rest are fixed by the GA. The empty brackets are so because it has not been considered any constraint. Finally, in the options it has been set the tolerance of the PS on function ( $Tolfun$ ) and on variable ( $TolX$ ).

### 8.1.6 Global minimum checking method

To develop this section a different solver from the standards ones supplied by MATLAB has been used. The solver used is included in the **OPTI Toolbox: NOMAD**, which uses a mesh adaptive direct search algorithm to solve non-differentiable and global

nonlinear programs. It also contains a facility to solve non-convex **mixed integer non-linear problems (MINLPs)**, and does this quite well for problems with high number of variable. Before using this solver, other solvers have been tested, specifically **GLPK** for **mixed integer linear programming (MILP)** an **SCIP** for **mixed integer quadratic programming (MIQP)**, but no one of these have been run successfully (both also included in OPTI Toolbox).

This is an example of this solver for the case of one DVA:

```
% Bounds
lb = [1; 1; 2; 0.5*10^7; 500];
ub = [39; 10; 40; 0.5*10^8; 1000];

xtype = 'IIICC';

%Initial Pointy
x0 = [1; 1; 1; 0.5e+07; 500];

% Options
opts = optimset('solver','nomad','display','iter')

Opt =
opti('fun',@fitness_1DVA,'bounds',lb,ub,'xtype',x
type,'options',opts)

[x,fval,exitflag] = solve(Opt,x0)
```

As it is possible to see this solver is quite easy to use and follows the same structure as the solvers explained before (GA and PS):

1. Set up the **bounds** and configure which type of number is each variable (**xtype**): 'IIICC' means that there are five unknown values the first three are integers (I is equal to integer) and the last two are continuous (C is equal to continuous).
2. Select a **starter point x0**: the lowest bound has been selected in order to not to approach directly to the optimal solution.
3. Configure the options **optimset**: select the solver ('nomad') and as in the GA the information to display in each iteration.
4. Finally generate the information of the solver (**OPTI object**) with the fitness function.

The advantage of using this algorithm is that it is very similar to the other algorithms presented in this study, so it is just needed to adapt the task done before to the specifications of NOMAD.

This code has been tested for the optimization of one DVAs and the result has been:

$$x = [32 \ 1 \ 9 \ 50000000 \ 1000]; \text{fval}=2.3105\text{e-}024;$$

As in the case of the GA each component of the vector means:

| Parameters             | Units                                             | Values      |
|------------------------|---------------------------------------------------|-------------|
| NodeID                 | -                                                 | 32          |
| DistributionProperties | m                                                 | 1           |
| Numberof_LD_DVA        | -                                                 | 9           |
| k                      | $\text{N}\cdot\text{m}^{-1}$                      | 5e+07       |
| c                      | $\text{N}\cdot\text{s}\cdot\text{m}^{-1}$         | 5e+05       |
| m                      | kg                                                | 1000        |
| fval                   | $\text{m}^2\cdot\text{s}^{-4}\cdot\text{Hz}^{-2}$ | 2.3105e-024 |

Table 8.11: Solutions of the solver NOMAD for the optimization for one DVA.

These results match with the solution which has been found using the GA (results of the GA will be presented in chapter 9. Results and analysis), this means that it has been found the same result by different ways so there is a high certainty that the solution corresponds to a global minimum. Then the model proposed to check the solutions by using an alternative solver is enough accurate as to give an approach if the solution is an optimal point.

## 9. Results and analysis

In this section the results obtained are presented and discussed. The GA has been executed for the cases of multiple DVAs, from 1 to 10. The solutions are presented in tables 9.1, 9.2 and 9.3. For each number of DVAs there is the complete information of all the DVAs placed for each configuration. The DVAs have been ordered by the position (NodeID) and the new DVA calculated in each configuration is highlighted in grey.

About the results, there are two different points: the quality of the solutions obtained and the behaviour of the algorithm. Although the quality of solutions obtained has to be analysed, it is important to understand that results have to demonstrate that the algorithm proposed is able to find the global minimums and also, the velocity of the algorithm has to be evaluated.

In this way, to evaluate the reduction of the vibration levels it has been used the **insertion loss** (IL), in **decibels** (dB). This parameter is useful in order to evaluate the loss of the response, resulting from the insertion of DVAs. The equation is given by

$$IL_n = 10 \log_{10} \left( \frac{PSD_0}{PSD_n} \right) \quad (9.1)$$

Where  $PSD_0$  is the power spectral density (calculated in section 8.1.2.3, see equation 8.8) for the system without DVAs and  $PSD_n$  for the system with  $n$  DVAs applied; for each number of DVAs it is obtained a different value of IL which is expected to be larger as the number of DVAs increases. This behaviour can be observed in the results, as shown in tables 9.1, 9.2 and 9.3. From an initial value of **0.9487 dB**, for one DVA, it increases progressively to **4.5596 dB** for ten DVAs. The LEAM (the research center in which this study has been developed), expected an IL inside the range between 4 dB to 6 dB, so the final solution agrees with the expectations.

About the other parameters, it is interesting to see that some of them tend to be near the bounds, it is the case of the **DistributionProperties** and the **mass**. The first one tends to the lower bound because in this way, the DVAs are placed along the longitudinal distance as near as it is possible. The second one, the mass tends to the upper bound due to that effectiveness of the DVAs increases with increasing mass, because a larger value of mass allows dissipating more energy by the DVAs. Other



interesting parameter is **Numberof\_LD\_DVA**; for the first configurations the value is the same, but while the number of DVAs is increased, this parameter changes, mainly increasing too. This means that for higher number of DVAs placed in the cross section, more DVAs are needed along the longitudinal distance in order to reach the optimal solution. Finally, about the positions in which the DVAs are placed (**NodeID**), it is quite interesting to see how, except the last one, all the locations are moving around (above and below) of the two first ones, 32 and 8.

To get a better understanding of the performances of the DVAs, a plot containing the solution for different cases has been presented. This plot, figure 9.1, represents the square of the acceleration, in absolute value, in front of the frequency for three different cases. The first one is for the case without DVAs, the second one, for one DVA and the last one for the maximum number of DVAs, ten DVAs. As it can be seen, there is a clear reduction of the vibration level by installing DVAs. In the first two cases, the final profile of the response is quite similar, that is because although there is an attenuation of the vibrations, to configure just one DVA is not enough as to change the shape of the profile. In this way, for three DVAs, it is clear the differences of the profiles in the plots. For most of the frequencies the levels have been really minimized and just the range from 20 to 60 Hz generates a more sharp profile. The plots for all the optimization cases are presented in annex C.

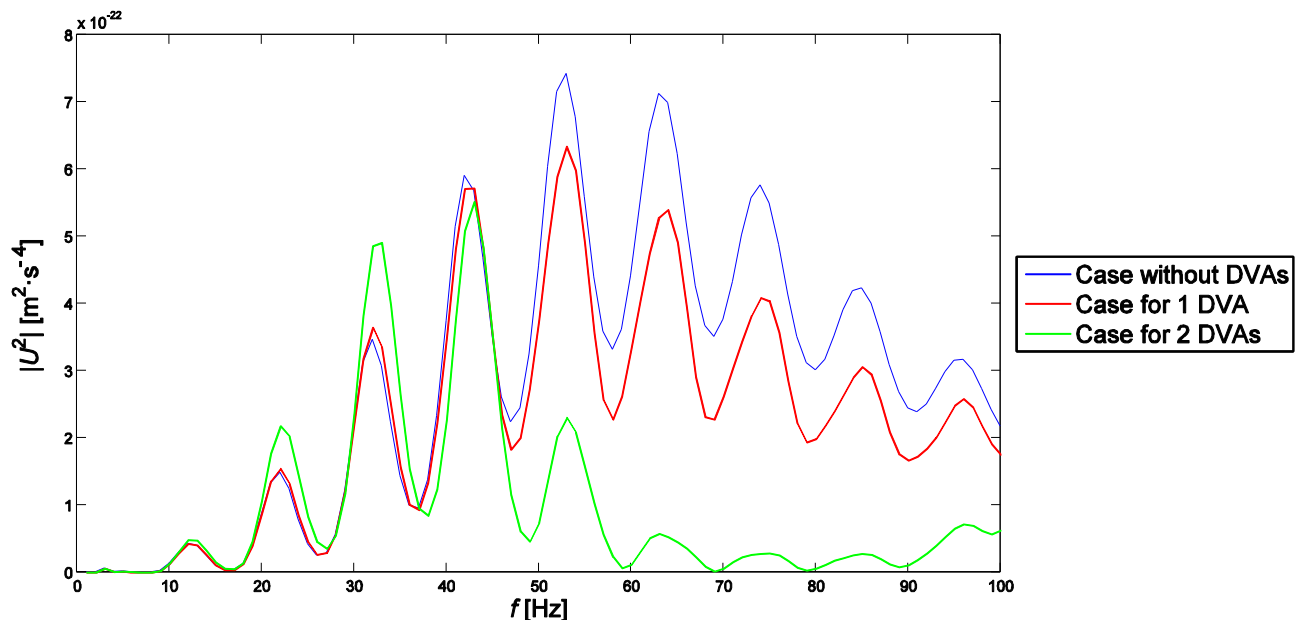


Figure 9.1: Graph of the square of the acceleration against frequency, for three cases: without DVAs, for one DVA and for ten.

About the mechanical behaviour of the DVAs, there is an important parameter to take in account, the **natural frequency**. At this frequency, the DVA tend to oscillate as much as it can, so the performance will be considerably better. In this way, it has been considered interesting to evaluate the natural frequency of the cases, for example for the case of the figure 9.2, for one DVA, if it is calculated the natural frequency as ( $\omega_0$  in  $\text{rad}\cdot\text{s}^{-1}$ ):

$$\omega_0 = \sqrt{\frac{k}{m}} \quad (9.2)$$

where  $k$  is the stiffness ( $\text{N}\cdot\text{m}^{-1}$ ),  $m$  is the mass (kg) and  $\omega_0$  is in  $\text{rad}\cdot\text{s}^{-1}$ . Using the values for one DVA of the table 9.1 it is obtained a  $\omega_0$  value of **223.60  $\text{rad}\cdot\text{s}^{-1}$** , which corresponds to a natural frequency of **35.587 Hz**. This value, as it can be seen in figure 9.1, does not correspond to the frequency in which attenuation of the vibrations is maximized. Mainly, there are three reasons or hypothesis which can justify this behaviour:

- The **damping**; the fact that the DVAs have been implemented with damping, generates a considerably reduction of the effect in the natural frequency and this effect is distributed by the rest of frequencies. This phenomenon is because the vibrations level of the DVA for its resonance frequency is reduced as the value of the damping increases, then the DVA is able to keep less energy at this specific frequency.
- The **distribution of the DVAs** along the longitudinal distance; the fact that the DVAs periodically affect to the waves which are circulating through the system, possibly it generates effects at frequencies different from the natural.
- Malfunction of the function **H\_UnitaryExteranlForce\_E1.m**.

The last comment is about the **computing time** of the algorithm. Along this study, the time needed by the algorithm to converge has been one of the priorities to manage. In order to evaluate this aspect, it has been measured the time taken by the algorithms to find a solution. Obviously, when the number of DVAs increases, the computing time of the GA also increases. For example, just to get an idea, the optimization for one DVA took **4.73 hours**, for two DVAs took **12.40 hours**, for three DVAs took **18.68 hours**, for four DVAs took **21.53 hours** and for five DVAs, **27.19 hours**. With these measures, it is clear that the computing time of the GA increases substantially by increasing the number of the DVAs, but what is important to be clear is that the convergence of the

## 9. Results and analysis

---

GA remains exactly the same for all the cases; the problem is the function `H_UnitaryExternalForce_E1.m`, which takes most of the computing time. Therefore GA, more or less, always stops in the same number of generations; the problem is that it is required more time to calculate the fitness value of the individuals. In this way, one of the points proposed as future lines of investigations is to improve the speed of this function.

## 9. Results and analysis

| Number of DVAs | NodeID | DistributionProperties | Numberof_LD_DVA | k [N·m <sup>-1</sup> ] | c [N·s·m <sup>-1</sup> ] | m [kg]  | IL [dB] |
|----------------|--------|------------------------|-----------------|------------------------|--------------------------|---------|---------|
| 1              | 32     | 1                      | 9               | 4.9998e+07             | 4.9998e+05               | 999.99  | 0.9     |
| 2              | 8      | 1                      | 9               | 4.9999e+07             | 4.9999e+05               | 1,000.0 | 1.8     |
|                | 32     | 1                      | 9               | 4.9998e+07             | 4.9998e+05               | 999.99  |         |
| 3              | 8      | 1                      | 9               | 4.9999e+07             | 4.9999e+05               | 1,000.0 | 2.4     |
|                | 32     | 1                      | 9               | 4.9998e+07             | 4.9998e+05               | 999.99  |         |
|                | 33     | 1                      | 11              | 3.6969e+07             | 3.6969e+05               | 998.17  |         |
| 4              | 8      | 1                      | 9               | 4.9999e+07             | 4.9999e+05               | 1,000.0 | 3.0     |
|                | 9      | 1                      | 10              | 4.9494e+07             | 4.9494e+05               | 992.90  |         |
|                | 32     | 1                      | 9               | 4.9998e+07             | 4.9998e+05               | 999.99  |         |
|                | 33     | 1                      | 11              | 3.6969e+07             | 3.6969e+05               | 998.17  |         |
| 5              | 8      | 1                      | 9               | 4.9999e+07             | 4.9999e+05               | 1,000.0 | 3.6     |
|                | 9      | 1                      | 10              | 4.9494e+07             | 4.9494e+05               | 992.90  |         |
|                | 31     | 1                      | 18              | 4.9744e+07             | 4.9744e+05               | 999.69  |         |
|                | 32     | 1                      | 9               | 4.9998e+07             | 4.9998e+05               | 999.99  |         |
|                | 33     | 1                      | 11              | 3.6969e+07             | 3.6969e+05               | 998.17  |         |
| 6              | 7      | 1                      | 14              | 2.2011e+07             | 2.2011e+05               | 991.87  | 3.9     |
|                | 8      | 1                      | 9               | 4.9999e+07             | 4.9999e+05               | 1,000.0 |         |
|                | 9      | 1                      | 10              | 4.9494e+07             | 4.9494e+05               | 992.90  |         |
|                | 31     | 1                      | 18              | 4.9744e+07             | 4.9744e+05               | 999.69  |         |
|                | 32     | 1                      | 9               | 4.9998e+07             | 4.9998e+05               | 999.99  |         |
|                | 33     | 1                      | 11              | 3.6969e+07             | 3.6969e+05               | 998.17  |         |

Table 9.1: Solutions from 1 to 6 DVAs.

## 9. Results and analysis

| Number of DVAs | NodeID | DistributionProperties | Numberof_LD_DVA | k [N·m <sup>-1</sup> ] | c [N·s·m <sup>-1</sup> ] | m [kg]  | IL [dB] |
|----------------|--------|------------------------|-----------------|------------------------|--------------------------|---------|---------|
| 7              | 7      | 1                      | 14              | 2.2011e+07             | 2.2011e+05               | 991.87  | 4.2     |
|                | 8      | 1                      | 9               | 4.9999e+07             | 4.9999e+05               | 1,000.0 |         |
|                | 9      | 1                      | 10              | 4.9494e+07             | 4.9494e+05               | 992.90  |         |
|                | 10     | 1                      | 17              | 2.8992e+07             | 2.8992e+05               | 997.11  |         |
|                | 31     | 1                      | 18              | 4.9744e+07             | 4.9744e+05               | 999.69  |         |
|                | 32     | 1                      | 9               | 4.9998e+07             | 4.9998e+05               | 999.99  |         |
|                | 33     | 1                      | 11              | 3.6969e+07             | 3.6969e+05               | 998.17  |         |
| 8              | 7      | 1                      | 14              | 2.2011e+07             | 2.2011e+05               | 991.87  | 4.3     |
|                | 8      | 1                      | 9               | 4.9999e+07             | 4.9999e+05               | 1,000.0 |         |
|                | 9      | 1                      | 10              | 4.9494e+07             | 4.9494e+05               | 992.90  |         |
|                | 10     | 1                      | 17              | 2.8992e+07             | 2.8992e+05               | 997.11  |         |
|                | 30     | 1                      | 18              | 2.1991e+07             | 2.1991e+05               | 998.99  |         |
|                | 31     | 1                      | 18              | 4.9744e+07             | 4.9744e+05               | 999.69  |         |
|                | 32     | 1                      | 9               | 4.9998e+07             | 4.9998e+05               | 999.99  |         |
|                | 33     | 1                      | 11              | 3.6969e+07             | 3.6969e+05               | 998.17  |         |
| 9              | 7      | 1                      | 14              | 2.2011e+07             | 2.2011e+05               | 991.87  | 4.5     |
|                | 8      | 1                      | 9               | 4.9999e+07             | 4.9999e+05               | 1,000.0 |         |
|                | 9      | 1                      | 10              | 4.9494e+07             | 4.9494e+05               | 992.90  |         |
|                | 10     | 1                      | 17              | 2.8992e+07             | 2.8992e+05               | 997.11  |         |
|                | 30     | 1                      | 18              | 2.1991e+07             | 2.1991e+05               | 998.99  |         |
|                | 31     | 1                      | 18              | 4.9744e+07             | 4.9744e+05               | 999.69  |         |
|                | 32     | 1                      | 9               | 4.9998e+07             | 4.9998e+05               | 999.99  |         |
|                | 33     | 1                      | 11              | 3.6969e+07             | 3.6969e+05               | 998.17  |         |
|                | 34     | 1                      | 18              | 1.5924e+07             | 1.5924e+05               | 998.91  |         |

Table 9.2: Solutions from 7 to 9 DVAs.

## 9. Results and analysis

| Number of DVAs | NodeID | DistributionProperties | Numberof_LD_DVA | k [N·m <sup>-1</sup> ] | c [N·s·m <sup>-1</sup> ] | m [kg]  | IL [dB] |
|----------------|--------|------------------------|-----------------|------------------------|--------------------------|---------|---------|
| 10             | 7      | 1                      | 14              | 2.2011e+07             | 2.2011e+05               | 991.87  | 4.7     |
|                | 8      | 1                      | 9               | 4.9999e+07             | 4.9999e+05               | 1,000.0 |         |
|                | 9      | 1                      | 10              | 4.9494e+07             | 4.9494e+05               | 992.90  |         |
|                | 10     | 1                      | 17              | 2.8992e+07             | 2.8992e+05               | 997.11  |         |
|                | 16     | 1                      | 24              | 3.7992e+07             | 3.7992e+05               | 999.00  |         |
|                | 30     | 1                      | 18              | 2.1991e+07             | 2.1991e+05               | 998.99  |         |
|                | 31     | 1                      | 18              | 4.9744e+07             | 4.9744e+05               | 999.69  |         |
|                | 32     | 1                      | 9               | 4.9998e+07             | 4.9998e+05               | 999.99  |         |
|                | 33     | 1                      | 11              | 3.6969e+07             | 3.6969e+05               | 998.17  |         |
|                | 34     | 1                      | 18              | 1.5924e+07             | 1.5924e+05               | 998.91  |         |

Table 9.3: Solutions for 10 DVAs.

## 10. Environmental and safety implications

Nowadays engineering is really concerned about environmental impact, most of the studies consulted along this study show a sustainable conscious to the environment. As it has been explained in chapter 5, there are two elements which generate environment effects: **noise** and **vibration**. Of course, for this case study, noise is generated as consequence of the induced vibration. In the following sections, the main environmental concerns derived from this study will be developed.

### 10.1 Acoustic reduction

Acoustic pollution is a recurrent problem which historically, has been tried to be solved by installing high barriers or sinking in track cuttings; however, costs are prohibitive. The European Union boosted in 2013 new projects to mitigate railway noise. According to the **European Environment Agency**, rail noise affects about 12 million EU inhabitants at day time, with a noise exposure above 55 dB, and about 9 million at night time, with a noise exposure above 50 dB [34].

From the past years, there have been some initiatives in this line. The **Environmental Noise Directive** (2002/49/EC) [35], the issue of noise caused by rail transport has gained increasing attention in the field of sustainable transport. In 2005 the Commission adopted the **Decision 2006/66/EC** [36] concerning the technical specification for interoperability relating to the noise of rolling stock (TSI Noise), and in 2008 the **Communication on rail noise abatement measures addressing the existing fleet**. However, due to the long lifetime of rolling stock, it is believed that it will take many years before the overall noise emissions from freight trains can be reduced significantly if no additional measures addressing the existing fleet are introduced.

This study could contribute to the block of measures and research in which the EU is supporting to achieve its objectives. In table 10.1 there are some other alternatives and studies which are being developed around the EU. These studies are being developed during the last years, trying to accomplish with objectives established by the European Union to mitigate noise because of train traffic. The aim is to make a change in how

traffic trains impacts into the environment and how this affects to near buildings and directly to the comfort and distress of the inhabitants.

| COUNTRY     | TOPIC                                        | DESCRIPTION                                                                                                                                                           |
|-------------|----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Netherlands | Shunting yards                               | Lubrication, removing rail joints, noise barriers and window insulation                                                                                               |
|             | Research projects                            | Friction modifiers against curve squeal, influencing rail roughness                                                                                                   |
|             | Management of noise ceilings                 | Monitoring noise ceilings and capacity management                                                                                                                     |
| Germany     | Testing innovative infrastructure measures   | Rail dampers, friction modification, low height barriers, absorbers for steel bridges, under sleeper pads                                                             |
|             | Work on realistic rail/wheel contact         | Improvement of wheel/rail contact, wheel vibration and acoustic optimization of pavement                                                                              |
| France      | Wheel and rail dampers                       | Combined optimization of rail and wheel dampers. Homologation of wheel dampers (STARDAMP project)                                                                     |
| Denmark     | Research and testing programmes              | Optimization of track construction, acoustic rail grinding, noise partnership with the inhabitants an noise communication management                                  |
| Switzerland | Additional measures                          | As cost-benefit analysis should show which additional measures will be taken: rail grinding, stand by noise, rail dampers, steel bridges are among the issues studied |
| Sweden      | Noise abatement programme and special topics | Acoustic grinding, tests of special measures such as rail absorbers and low height barriers                                                                           |
| Norway      | Research and tests                           | Rail grinding planned but not yet implemented, nose from height terminals, tonal noise from accelerating and decelerating trains                                      |

Table 10.1: Examples of further solutions being implemented in various countries of the EU [9].

## 10.2 Building damage reduction

An important problem associated is that ground-borne vibration can cause noticeable vibrations in near buildings. Householders are normally worried that vibrations might damage their property. For that reason along the study the final vibration level is



## 10. Environmental implications

analysed in a point, which represents a position inside a building. So it is possible to evaluate the vibration transmitted to the near buildings.

There are three components to any vibration system: the **source**, the **transmission path** and the **receiver** (structure/building). This study actuates in the second component, the transmission path, with the application of the DVAs. Although vibrations induced in buildings by ground-born excitation are often noticeable, there is little evidence that they produce even a cosmetic damage, for example small cracks in plaster. So in case of damage concern may occur, it is important to survey the property before exposure to the vibrations, measure the vibration levels induced and finally check if any damage is evident. Some of the parameters needed to be measured are:

- **Particle velocity:** velocity of particles set into motion by the propagation of a disturbance through the ground and a structure by a source of vibration.
- **Frequency:** the maximum instantaneous particle velocity at a point during the given time interval.
- **Peak component particle velocity:** the maximum value of any of the three orthogonal component particle velocities measured during time interval.

| Line<br><i>see Fig 1</i> | Type of building                                                                           | Peak component particle velocity<br>in frequency range of predominant pulse |                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|--------------------------------------------------------------------|
|                          |                                                                                            | 4 Hz to 15 Hz                                                               | 15 Hz and above                                                    |
| 1                        | Reinforced or framed structures<br>Industrial and heavy commercial<br>buildings            | 50 mm/s at 4 Hz and above                                                   |                                                                    |
| 2                        | Unreinforced or light framed<br>structures<br>Residential or light commercial<br>buildings | 15 mm/s at 4 Hz<br>increasing to<br>20 mm/s at 15 Hz                        | 20 mm/s at 15 Hz<br>increasing to<br>50 mm/s at 40 Hz<br>and above |

*Values are at the base of the building.*

*For line 2, at frequencies below 4 Hz, a maximum displacement of 0.6 mm (zero to peak) should not be exceeded.*

Figure 10.1: Transient vibration guide values for damage, extracted BS 7385: Part 2 [8].

In figure 10.1, there are shown the maximum values of some of the parameters measured, allowed as vibration level received, based on the normative. In conclusion, by reducing vibration levels, is possible to avoid all this process because by normative with very low levels it is not necessary to make an evaluation of the property. In this way, future environmental issues derivative from the damages in buildings are avoided.

### 10.3 Energy cost reduction

Energy is an important engineering problem nowadays. Trying to reduce or mitigate the energy cost of the algorithm is a good understanding of the **efficiency** of the code. One of the objectives of this study is to generate an algorithm, a methodology to be able to solve the problem in a reasonable amount of time.

In this line consumption of energy is also critical in computer systems, in terms of **cost** and **availability**. From electricity costs impose a substantial strain on the budget of data and computing centers, also in office environments, computers and monitors account for the highest energy consumption after lighting.

**Power dissipation** is also a major concern in portable, battery-operated devices. It is common that batteries of laptop are depleted quickly. Energy dissipation also causes **thermal problems**. Most of the energy consumed by a system is converted into heat, resulting in wear and reliability of hardware components [37]. In this way all the methods presented in this work with the aim of improving the performance of the algorithm because saving time is equivalent to save energy and they are a good way to develop a sustainable treatment with the environment.

## 11. Future lines of investigation

The future lines of investigation of this study go mainly, in two directions. The first idea is to generate a more realistic approach to the problem, by introducing elements to the system as the moving load and the inclusion of a building. The second point is about the algorithm; the study of how to improve it and the study of the effect of changes in GA parameters in terms of efficiency.

The first part, studies a more realistic model, including two main items:

- Study the **moving load** case: this is the load that changes in time the place to which it is applied. This phenomenon is associated to the train passing on the tracks and by including it in the problem, the complexity increases substantially.
- Inclusion of a **building**: this part is important in order to see the differences into the response associated to the ground and to the building. The mechanical importance of the building remains in the fact that the building is considered as a **finite system** (not as the ground) and it presents a very strong **modal behaviour**.

About this first part, it will also be interesting to extend the code by using finite elements and study the response of the system, as well as the behaviour of the DVAs by introducing **discrete distributions** of the DVAs along the longitudinal distance. What is expected from this evaluation is to obtain better results; the system **tunnel—ground—building**, generates a sharper spectrum which is more suitable for the DVAs. That is because with a sharp profile the DVAs will be clearly tuned for specific frequencies, so at the attenuation levels will be higher.

The second part of the future lines of investigation is related to the algorithm in terms of **refinement** and **speed up** the code. It is needed a deeper study of how parameters of the GA affect its efficiency: mutation, reproduction, elitism... These analyses require better hardware than the one which has been used in the current study. Furthermore, to be able to improve the performance of H\_UnitaryExternalForce\_E1.m it is also required an improvement of the hardware, in case of need, for sure a cluster will be a good option because it will ensure the improvement of the parallel options.

## 12. Timeline

This chapter presents the tasks and the timeline proposed to continue the study for the futures lines of investigation. Figure 12.1 represents the timeline model proposed to cover the continuation of this study. The complete information of the planning and scheduling of both studies, the current and the future one, and the interdependencies of the tasks are presented in annex D. The following list contains a brief description of the tasks and work packages that must be developed to study the future lines of investigation:

**A. Information review and research:** first approach of the moving load problems as well as to understand how to include the building in the system. Furthermore, to investigate new ways to speed up the code.

**A.1 Review railway-induced vibration problem with moving load:** review the basics of moving load applied to railway-induced vibration problem.

**A.2 Review how to introduce the building in the system:** search for the procedure of how to couple DVAs to tunnel walls.

**A.3 Genetic algorithm programming research:** research of how to implement a more complex and accurate refinement of the GA and look for alternative ways to improve the performance.

**B. Identification of the problem:** evaluation of the problem (case study) to solve and the functions/equations to be optimized.

**C. Theoretical background:** all the theoretical concepts needed to evaluate the problem.

**C.1 Moving load theoretical description:** theoretical background needed to be able to set the problem.

**C.2 Inclusion of the building as a part of the system:** develop the theoretical concepts to be able to introduce this part to the global system.

**C.3 Description of the dynamic response of the train:** theoretical background needed to be able to set the problem.

**D. Optimization process:** development of the optimization procedure.

**D.1 Formulation of the problem:** formulation of the optimization problem for the different study cases and for the global problem.

**D.2 Study of the performance of each part of the algorithm:** study of each component of the algorithm and how to speed it up.

**D.2.1 Implementation of the algorithm:** programming the complete algorithm for the study case included the genetic algorithm.

**D.2.2 Study the effects of mutation strategy:** study of the effects generated by changing the mutation ratio.

**D.2.3 Study the effects of crossover strategy:** study of the effects generated by changing the crossover ratio.

**D.2.4 Study the effects of elitism strategy:** study of the effects generated by changing the elitism ratio.

**D.2.5 Study of the improvement of H\_UnitaryExternal Force\_E1.m:** study of how to improve this function in order to speed up the code.

**D.3 Optimization case:**

**D.3.1 Implementation of the algorithm:** programming the complete algorithm included the genetic algorithm.

**D.3.2 Final refinement of the genetic algorithm:** after all the improving methods, at least with the final code, it is needed to evaluate the GA parameters in order to allow more accurate treatment.

**D.3.3 Validation of the algorithm:** correlation of the results with data provided by the research group.

## 12. Timeline

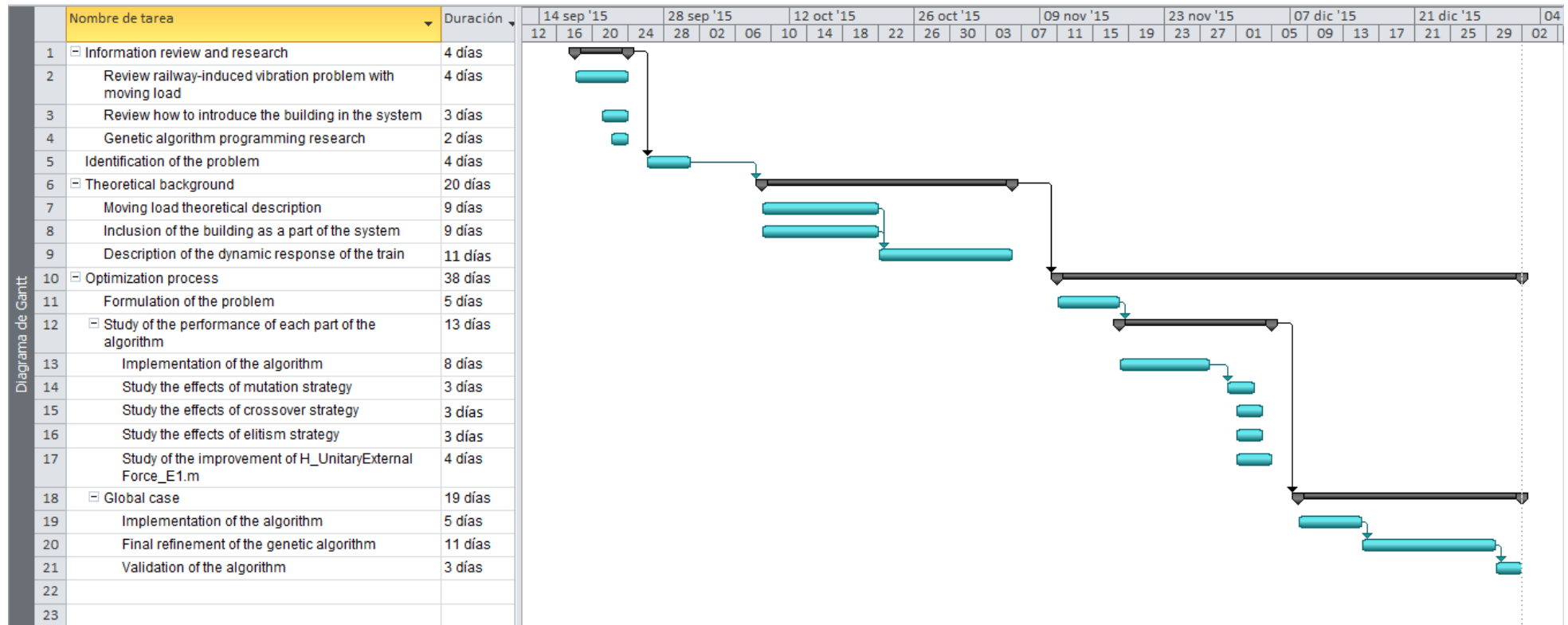


Figure 12.1: Gantt chart of the future study. It has been considered that it starts on September 18<sup>th</sup> and finishes on January 1<sup>st</sup>.

## 13. Budget

The budget is covered in detail in a separate document. This section presents an overview of the final cost of the study, covering the four main expenses:

- **Labour** (human costs), expenses related to the hours worked by the corresponding engineer. Total cost of 16,200 €.
- **Hardware**, expenses related to the costs of the hardware used to develop this study as well as to run the code. Total cost of 1,200 € for a two years period (200 €).
- **Software**, expenses related to the costs of purchasing the software used to develop this study, mainly MATLAB and the corresponding *Toolbox*. Total cost of 5,150 € for a one year period (1,716.6 €).
- **Energy**, expenses related to the energy consumption. Total cost of 15.6 €.

The final cost of the study is 18,132.2 €.

## 14. Economic viability

In this section, an estimation of the costs resulted from installing the solutions obtained into the tunnel is presented. First, each distribution of DVAs has been placed in its corresponding position (NodeID) of the tunnel, figure 1.1. From this figure, two different types of installation are defined. The first one, for NodeIDs 7 to 10 and 30 to 34, it will be an system which combines a **concrete block**, a **spring** and a **metallic platform** to fix the DVA in the tunnel wall. The second type is for NodeID 16 and is simpler than the first one, just a concrete block and **elastomeric material**, because it will be fixed directly over the floor of the railway.

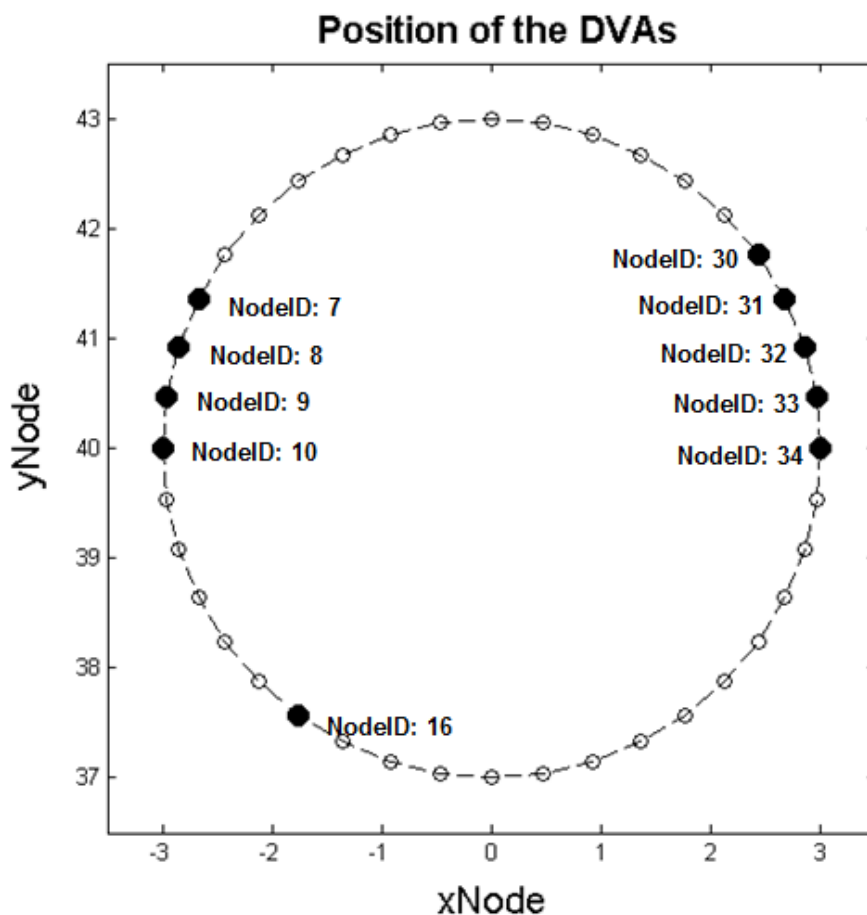


Figure 14.1: Position of each distribution of DVAs. xNode and yNode correspond to the coordinates of the Evaluators (see annex B) .



In table 1.1 the total costs of the materials and components needed to build and to install the DVAs are presented. It has been chosen a high strength concrete from CEMEX with a density of  $2,400 \text{ kg/m}^3$  and a price of  $120 \text{ €/m}^3$ . The mass of the blocks of concrete corresponds to the mass obtained for each DVA. The elastomeric material is a type of rectangular and laminated elastomeric bearing and after reviewing different resources it has been selected as a reference price  $140.00 \text{ €}$  for each sheet. The springs are supplied by SODEMANN, with a cost of  $710.48 \text{ €}$  for all the springs ( $71.050 \text{ €}$  each one); of course for the different DVAs distribution springs have specific mechanical properties according to the results obtained. Finally, the metallic platform has been estimated with a price of  $225.00 \text{ €}$  per unit.

| NodeID | Number of DVAs | Mass [kg] | Cost [€] |                      |        |                   |        |                    |
|--------|----------------|-----------|----------|----------------------|--------|-------------------|--------|--------------------|
|        |                |           | Concrete | Elastomeric material | Spring | Metallic platform | DVA    | Total distribution |
| 7      | 14             | 991.87    | 49.600   | -                    | 71.050 | 225.00            | 345.65 | 4,839.1            |
| 8      | 9              | 1,000.0   | 50.000   | -                    | 71.050 | 225.00            | 346.05 | 3,114.5            |
| 9      | 10             | 992.90    | 49.600   | -                    | 71.050 | 225.00            | 345.65 | 3,456.5            |
| 10     | 17             | 997.11    | 49.900   | -                    | 71.050 | 225.00            | 345.95 | 5,881.2            |
| 16     | 24             | 999.00    | 50.000   | 140.00               | -      | -                 | 190.00 | 4,560.0            |
| 30     | 18             | 998.99    | 49.900   | -                    | 71.050 | 225.00            | 345.95 | 6,227.1            |
| 31     | 18             | 999.69    | 50.000   | -                    | 71.050 | 225.00            | 346.05 | 6,229.9            |
| 32     | 9              | 999.99    | 50.000   | -                    | 71.050 | 225.00            | 346.05 | 3,114.5            |
| 33     | 11             | 998.17    | 49.900   | -                    | 71.050 | 225.00            | 345.95 | 3,805.5            |
| 34     | 18             | 998.91    | 49.900   | -                    | 71.050 | 225.00            | 345.95 | 6,227.1            |

Table 14.1: Total cost of the materials and components of each distribution of DVAs.

The final cost of the DVAs is **47,455.4 €**. The human costs have been estimated as a 40 % of the total cost ( $31,636.9 \text{ €}$ ), so the total cost is **79,092.3 €**. This final value is just to have a reference of how much money would cost to bring to reality solutions obtained in this study and as an simple estimation there some costs that have not been included, for example the costs associated to stop train traffic during the works or the energy expenses. Finally, it is possible to obtain the cost for each decibel attenuated by dividing the final cost per the final insertion loss, **16,828.1 €/dB**.

## 15. Conclusions

The study has been completed successfully. The functions (DVAParameters.m and H\_UnitaryExternalForce\_E1.m) provided by the LEAM have been able to obtain the vibration levels in the evaluators by developing the railway-induced vibration problem. By using these functions, it has been implemented an optimization algorithm which from all the possibilities, it has been chosen a **genetic algorithm** (GA), because it allows working with **mixed integer problems**. In this way, as a part of the developing process of the GA, it has been used Optimization Toolbox of MATLAB to implement successfully the algorithm. Therefore, requirements (A), (B) and (C) from chapter 3, have been fulfilled.

Apart from developing the GA, it was required to adapt the algorithm for global optimization case. In order to achieve this requirement it has been implemented and tested a strategy to find the optimal solution for cases with more than one DVA. This strategy allows obtaining the optimal solution for each configuration, just by calculating one new DVA each time and using the previous calculated DVAs as known inputs. Then, it is also fulfilled requirement (D) and thanks to this strategy, as well as the methods implemented to improve the performance of the algorithm, it has developed a computationally cheap enough algorithm so the last requirement, (E), has been fulfilled.

About the algorithm, it was important to set up a method to improve the accuracy of the GA, as well as to reduce the charge of computing time in it. Because of that, it has been implemented a **hybrid algorithm**, combining the GA with a **pattern search** algorithm (PS). Furthermore, it has also been implemented also a model to check the solution, in order to confirm or ensure with higher certainty, those solutions obtained are really global minimums; for this case it has been used **NOMAD**, a **mixed integer non-linear solver**. So at the end, the final algorithm presents a modular structure that can be easily adapted to obtain better performances and to check the solutions, for future studies.

The last comment is about the performance of the DVAs and the future lines of investigation. As it has been commented previously, for future studies the calculation of the vibration levels will be performed in the building, using a proper mechanical model of it, and considering a moving train excitation. The receptances between the rail and the building are expected to be sharper than the ones between the rail and the ground surface used in this study, so clearly the DVAs will be tuned easily for these

## 15. Conclusions

frequencies and insertion loss is expected to be larger. Then, it is expected to obtained better results and it can be concluded that the application of DVAs into the tunnel problem is an interesting solution and in this way, the optimization code developed in this study is useful in order to solve it. However, it is needed to review H\_UnitaryExternalForce\_E1.m in order to confirm these predictions; in this review the behaviour of the DVAs as well as the reduction for the insertion loss have to be checked. The first point, because up to now the behaviour shown by the DVAs has been different from the expected so it has to be studied in detail and the second part is to confirm what has been explained before, that the DVAs will provide larger attenuation levels in presence of sharper receptances. Finally, it is also important to speed up this function to obtain more accurate solutions, without increasing overly computing.

## 16. Bibliography

- [1] Sutil, Tania. Las vibraciones al paso del tren por Valorio persisten debido al retraso de las obras. *La Opinión de Zamora* (2014).
- [2] Carlsson, Ulf. *Noise and vibration aspects on railway goods transportation*. Stockholm, Sweden: Järnvägsgruppen KTH (2003). Report 0506E.
- [3] S. Lakusic, M. Ahac. *Rail Traffic Noise and Vibration Mitigation Measures in Urban Areas*. Tehnički vjesnik (2012), Vol. 19, pp. 427-435. ISSN 1330-3651.
- [4] J. Maes, H. Sol. *A double tuned rail damper-increased damping at the two first pinned-pinned frequencies*. s.l. : Elsevier Ltd., (May 2003), Journal of Sound and Vibration, Vol. 267, pp. 721-737.
- [5] S. Naresh kumar, S. Gunasekharan. *Design and analysis of dynamic vibration absorber (DVA) to reduce vibration in rails*. EAAS & ARF, (April 2013), International Journal of Engineering and Applied Sciences, Vol. 3. ISSN 2305-829.
- [6] Yu Du, Ricardo A. Burdisso, Efstratios Nikolaidis. *Control of internal resonances in vibration isolators using passive and hybrid dynamic vibration absorbers*. Journal of Sound and Vibration (September 2005), Vol. 286, pp. 697-727.
- [7] Silviu Nastac, Adrian Lepona. *Structural Optimization of Vibration Isolation Devices for High Performances*, International Journal of Systems Applications, Engineering & Development (2008), Vol. 2.
- [8] Cheng Yang, Deyu Li, Li Cheng. *Dynamic vibration absorbers for vibration control within a frequency band*. Journal of Sound and Vibration (April 2011), Vol. 330, pp. 1582-1598.
- [9] Bianchi, Leonora. *A survey on metaheuristics for stochastic combinatorial optimization*. Natural Computing: an international journal (2009), Vol. 8, pp. 239–287.

- [10] Qiu, Z. *Genetic algorithm based active vibration control for a moving flexible smart beam driven by a pneumatic rod cylinder.*, Journal of Sound and Vibration (May 2012), Vol. 331, pp. 2233–2256.
- [11] Marano, Sara Sgobba and Giuseppe Carlo. *Optimal Design Criteria for Isolation Devices in Vibration Control, Stochastic Control*. InTech (2012). ISBN: 978-953-307-121-3.
- [12] Kaveh, A. and Rad, S. Malakouti. *Hybrid genetic algorithm and particle swarm optimization for the force method-based simultaneous analysis design*. B1, The Islamic Republic of Iran (2010), Iranian Journal of Science & Technology, Transaction B: Engineering, Vol. 34, pp. 15-34.
- [13] Matas, Edgar. *Study of optimization for vibration absorbing devices applied on airplane structural elements*. UPC (2014).
- [14] Alsedà, Dídac Sellés. *Study of coupling of vibration countermeasures applied on airplane structural elements*. UPC (2014).
- [15] MATLAB: [http://uk.mathworks.com/company/?s\\_tid=gn\\_co](http://uk.mathworks.com/company/?s_tid=gn_co), (consulted: March 2015).
- [16] Global Optimization Toolbox User's Guide: <http://uk.mathworks.com> (consulted: February 2015).
- [17] Parallel Computing Toolbox User's Guide: <http://uk.mathworks.com/products/parallel-computing/> (consulted: March 2015).
- [18] OPTI Toolbox: <http://www.i2c2.aut.ac.nz/Wiki/OPTI/index.php/Main/WhatIsOPTI> (consulted: March 2015).
- [19] Hunt, J. A. Forrest and H. E. M. *A three-dimensional tunnel model for calculation of train-induced ground vibration*. Journal of Sound and Vibration (July 2006), Vols. 94(4-5):678-705.
- [20] Hunt, M. F. M. Hussein and H. E. M. *A numerical model for calculating vibration from a railway tunnel embedded in a full-space*. Journal of Sound and Vibration ( August 2007), Vols. 305(3):401-431.

- [21] X. Sheng, C. J. C. Jones, and D. J. Thompson. *Prediction of ground vibration from trains using the wavenumber*. Journal of Sound and Vibration (June 2006), Vols. 293(3-5):575-586.
- [22] S. Gupta, M. F. M. Hussein, G. Degrande, H. E. M. Hunt, and D. Clouteau. *A comparison of two numerical models for the prediction of vibrations from underground railway traffic*. Soil Dynamics and Earthquake Engineering (July 2007), Vols. 27(7):608-624.
- [23] Forrest, J. A. *Modelling of ground vibration from underground railways*. PhD thesis, University of Cambridge (1990).
- [24] Kausel, A. J. B. Tadeu and E. *Green's functions for two-and-a-half-dimensional elastodynamic problems*. Journal of Engineering Mechanics - ASCE (2000) Vols. 126(10):1093-1097.
- [25] *Struct options*: <http://es.mathworks.com/help/matlab/ref/struct.html> (consulted: May 2015).
- [26] Tursa, James. *MTIMESX - Fast Matrix Multiply with Multi-Dimensional Support*:  
<http://www.mathworks.com/matlabcentral/fileexchange/25977-mtimesx-fast-matrix-multiply-with-multi-dimensional-support/content/mtimesx.m> (consulted: March 2015).
- [27] Luong, Bruno. *MultiSolver (M, RHS): Multiple same-size linear solver*:  
<http://www.mathworks.com/matlabcentral/fileexchange/24260-multiple-same-size-linear-solver/content/MultiSolverFolder/MultiSolver.m> (consulted: May 2009).
- [28] ISO 2631-2, Mechanical vibration and shock — Evaluation of human exposure to whole wholebody Part 2: Vibration in buildings (1 Hz to 80 Hz) (2003).
- [29] *Parallel Pools*: <http://es.mathworks.com/help/distcomp/parallel-pools.html> (consulted: April 2015).
- [30] *HWMonitor CPUi*:  
<http://www.cpuid.com/softwares/hwmonitor.html> (consulted: April 2015).

- [31] Intel® Core™ i7-2670QM Processor: [http://ark.intel.com/products/53469/Intel-Core-i7-2670QM-Processor-%286M-Cache-up-to-3\\_10-GHz%29](http://ark.intel.com/products/53469/Intel-Core-i7-2670QM-Processor-%286M-Cache-up-to-3_10-GHz%29) (consulted: April 2015).
- [32] Railway Noise Mitigation. *European Parliamentary Research Service*. European Parliamentary Research Service (August 2013): <http://epthinktank.eu/2013/08/29/railway-noise-mitigation/> (consulted: May 2015).
- [33] *Directive 2002/49/EC of the European Parliament and of the Council relating to the assessment and management of environmental noise*. Official Journal of the European Communities (2002).
- [34] *Commission Decision concerning the technical specification for interoperability relating to the subsystem rolling stock*. Official Journal of the European Union (2005).
- [35] Jakob Ortli, Peter Hübner. *Railway noise in Europe. A 2010 report in the state of the art*. Paris: International Union of Railways, UIC (2010).
- [36] *Damage to structures*. Garston: Construction Research Communications Ltd. for the Building Research Establishment (1995) .
- [37] Albers, Susanne. *Energy-Efficient Algorithms*. Communications of the ACM (2010), Vol. 53.

